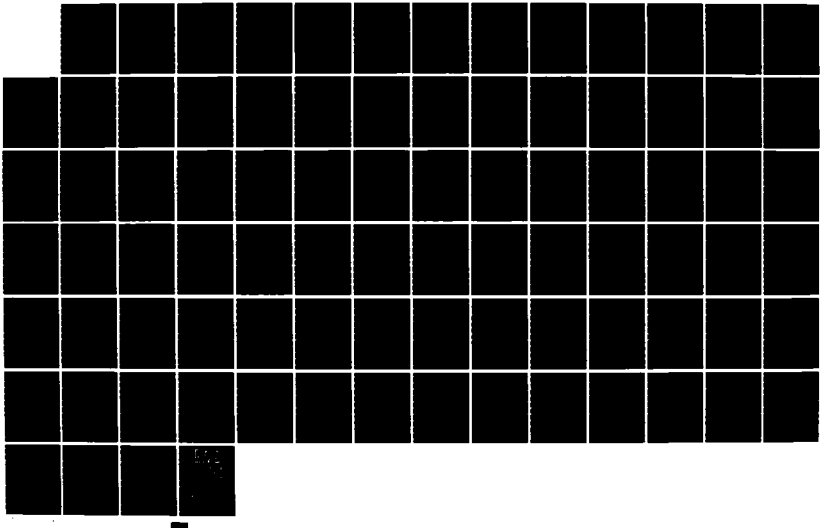
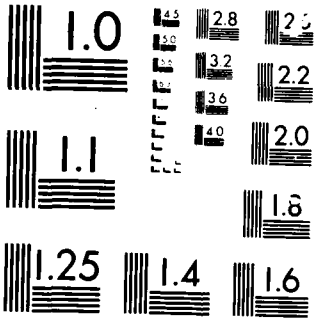
AD-A166 029 INTEGRATED BATTLEFIELD EFFECTS RESEARCH FOR THE NATIONAL TRAINING CENTER..(U) SCIENCE APPLICATIONS INTERNATIONAL CORP LA JOLLA CA D ERICKSON ET AL. 1/1

UNCLASSIFIED 31 DEC 84 SAIC-R-LJF-84-019-APP-I F/G 5/9 NL





MICROCOPY RESOLUTION TEST CHART

DNA-TR-85-13-AP-I

INTEGRATED BATTLEFIELD EFFECTS RESEARCH FOR THE NATIONAL TRAINING CENTER

Appendix I—Feasibility Study of Transferring ARTBASS Code from a Perkin-Elmer/Lexidata System to a VAX/De Anza System

Science Applications International Corporation
P. O. Box 2351
La Jolla, CA 92038-2351

31 December 1984

Technical Report

CONTRACT No. DNA 001-81-C-0273

THIS WORK WAS SPONSORED BY THE DEFENSE NUCLEAR AGENCY
UNDER RDT&E RMSS CODE S400082466 V99QAXNL00125 H2590D.

Prepared for
Director
DEFENSE NUCLEAR AGENCY
Washington, DC 20305-1000

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY N/A since Unclassified	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE N/A since Unclassified	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) R:LJF-84-019	5 MONITORING ORGANIZATION REPORT NUMBER(S) DNA-TR-85-13-AP-I
6a NAME OF PERFORMING ORGANIZATION Science Applications International Corporation 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Director Defense Nuclear Agency
6c ADDRESS (City, State, and ZIP Code) P.O. Box 2351 La Jolla, CA 92038-2351	7b ADDRESS (City, State, and ZIP Code) Washington, DC 20305-1000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER DNA 001-81-C-0273
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 62715H; PROJECT NO: V99QAXN; TASK NO: L; WORK UNIT ACCESSION NO: DH065313

11 TITLE (Include Security Classification)
INTEGRATED BATTLEFIELD EFFECTS RESEARCH FOR THE NATIONAL TRAINING CENTER
Appendix I—Feasibility Study of Transferring ARTBASS Code from a Perkin-Elmer/Lexidata

12 PERSONAL AUTHOR(S)
Erickson, D.; Ickler, J.; McKeown, P.; Metzger, L.; Plock, R.; Packard, B.; and Birney, J.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year Month Day)	15 PAGE COUNT
Technical	FROM 830613 TO 841230	841231	88

16 SUPPLEMENTARY NOTATION
This work was sponsored by the Defense Nuclear Agency under RDT&E RMSS Code S400082466 V99QAXNL00125 H2590D.

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP
5	9	
15	7	

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Training
Integrated Battlefield
Military Strategy
Military Doctrine

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Research performed to evaluate and develop enhancements for integrated battlefield training at the U.S. Army National Training Center is described. These enhancements had been identified and concepts developed for their application in earlier phases of this research. The report consists of the basic volume summarizing the research tasks, approach, results, conclusions, and recommendations; plus twelve appendices which provide details on the nine major tasks into which the research was divided. Research performed and the associated appendices are as follows:

Development of nuclear and chemical environmental and effects software:
- Analysis of nuclear algorithms — Appendix A
- Requirements specification for nuclear and chemical model algorithms at the NTC — Appendix B
- Chemical model algorithm description — Appendix C

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Betty L. Fox	22b TELEPHONE (Include Area Code) (202) 325-7042
	22c OFFICE SYMBOL DNA/STTI

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

11. TITLE (Continued)

System to a VAX/De Anza System

19. ABSTRACT (Continued)

Demonstration of the system for combining live and notional battalions for training higher level staffs in integrated battlefield (IB) command and control:

Functional requirements analysis for IB command and control simulation	Appendix D
Report on the demonstration	Appendix E

Analysis and design of field simulators for nuclear and chemical warfare:

Technical and operational impacts of field simulators	Appendix F
Capability of off-the-shelf paging system to communicate at Ft. Irwin	Appendix G
Designs of field simulators	Appendix H

Adaptation of nuclear and chemical software to other Army training models:

Feasibility of transferring ARTBASS Code from Perkin-Elmer to VAX	Appendix I
Division/Corps training simulation functional analysis	Appendix J
ARTBASS conversion to VAX	Appendix K
Requirements specification for adding nuclear and chemical models to ARTBASS	Appendix L

This research provided the following products:

Software which models nuclear and chemical environment and effects with appropriate fidelity and timing for training and which is ready for installation on NTC computers.

A demonstrated capability for combining actions of real battalions with computer simulated notional battalions for training brigade/division commanders and staffs.

An analysis of the impacts of using field simulators at the NTC for nuclear and chemical warfare training, and the designs of the selected simulators (i.e., common control system, radiacmeters, dosimeters, chemical detectors).

Analysis of the application of nuclear and chemical models to other Army battalion training models; conversion of the ARTBASS model to operate on the VAX 11/780; incorporation of the nuclear and chemical models into ARTBASS; and demonstration of the nuclear and chemical models using ARTBASS.

CONVERSION FACTORS FOR U.S. CUSTOMARY TO METRIC (SI) UNITS OF MEASUREMENT

To Convert From	To	Multiply By
angstrom	Meters (m)	1.000 000 x E -10
atmosphere (normal)	Kilo pascal (kPa)	1.013 25 X E +2
bar	kilo pascal (kPa)	1.000 000 X E +2
barn	meter^2 (m^2)	1.000 000 X E -28
British thermal unit (thermochemical)	joule (J)	1.054 350 X E +3
cal (thermochemical)/cm^2	mega joule/m^2 (MJ/m^2)	4.184 000 X E -2
calorie (thermochemical)	joule (J)	4.184 000
calorie (thermochemical)/g	joule per kilogram (J/kg)*	4.184 000 X E +3
curie	giga becquerel (Gbq) †	3.700 000 X E +1
degree Celsius	degree kelvin (K)	$t_K = t°_C + 273.15$
degree (angle)	radian (rad)	1.745 329 X E -2
degree Farenheit	degree kelvin (K)	$t_K = (t°_F + 459.67)/1.8$
electron volt	joule (J)	1.602 19 X E -19
erg	joule (J)	1.000 000 X E -7
erg/second	watt (W)	1.000 000 X E -7
foot	meter (m)	3.048 000 X E -1
foot-pound-force	joule (J)	1.355 818
gallon (U.S. liquid)	meter^3 (m^3)	3.785 412 X E -3
inch	meter (m)	2.540 000 X E -2
jerk	joule (J)	1.000 000 X E +9
joule kilogram (J/kg) (radiation dose absorbed)	gray (Gy)*	1.000 000
kilotons	terajoules	4.183
kip (1000 lbf)	newton (N)	4.448 222 X E +3
kip/inch^2 (ksi)	kilo pascal (kPa)	6.894 757 X E +3
ktap	newton-second/m^2 ($N-s/m^2$)	1.000 000 X E +2
micron	meter (m)	1.000 000 X E -6
mil	meter (m)	2.540 000 X E -5
mile (international)	meter (m)	1.609 344 X E +3
ounce	kilogram (kg)	2.834 952 X E -2
pound-force (lbf avoirdupois)	newton (N)	4.448 222
pound-force inch	newton-meter (N·m)	1.129 848 X E -1
pound-force/inch	newton/meter (N/m)	1.751 268 X E +2
pound-force/foot^2	kilo pascal (kPa)	4.788 026 X E -2
pound-force/inch^2 (psi)	kilo pascal (kPa)	6.894 757
pound-mass (lbm avoirdupois)	kilogram (kg)	4.535 924 X E -1
pound-mass-foot^2 (moment of inertia)	kilogram-meter^2 ($kg·m^2$)	4.214 011 X E -2
pound-mass/foot^3	kilogram-meter^3 (kg/m^3)	1.601 846 X E +1
rad (radiation dose absorbed	gray (Gy)*	1.000 000 X E -2
roentgen	coulomb/kilogram (C/kg)	2.579 760 X E -4
shake	second (s)	1.000 000 X E -8
slug	kilogram (kg)	1.459 390 X E +1
torr (mm Hg, 0° C)	kilo pascal (kPa)	1.333 22 X E -1

*The gray (Gy) is the accepted SI unit equivalent to the energy imparted by ionizing radiation to a mass and corresponds to one joule kilogram.

†The becquerel (Bq) is the SI unit of radioactivity; 1 Bq = 1 event s.

TABLE OF CONTENTS

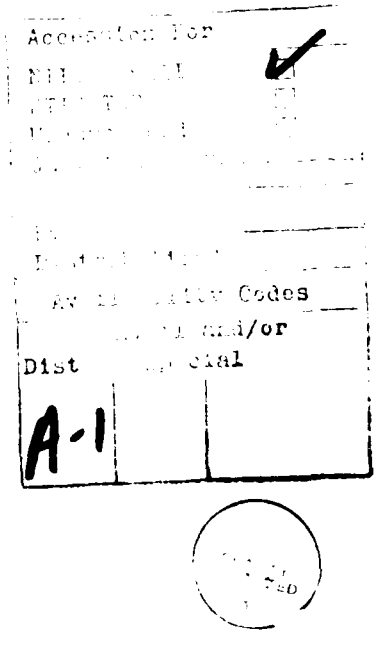
TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Concluded)

LIST OF ILLUSTRATIONS



SECTION 1
INTRODUCTION

The purpose of this document is to delineate those items involved in the creation of an ARTBASS-M development system which will be based on a VAX 11/780 system. This includes the following items:

1. The physical transfer of ARTBASS-M code currently resident on the Perkin-Elmer system to a permanent development system based on the VAX 11/780.

2. The conversion of Perkin-Elmer based FORTRAN 77 code to VAX FORTRAN 77.

3. The implementation of the working ARTBASS-M system on the VAX.

4. The subsequent transfer of the working VAX ARTBASS-M code back to the Perkin-Elmer system.

5. The installation and implementation of the working ARTBASS-M system on the Perkin-Elmer system.

Each of these aspects will be discussed in the following sections.

In order to fully understand the ensuing discussion, the underlying assumptions of this study should be enumerated. They are the following:

1. There will be no modifications to the ARTBASS-M code that currently exists on the Perkin-Elmer system.

2. Only those lines of code that do not or cannot execute on the VAX system will be altered to make the Perkin-Elmer ARTBASS-M code resident on the VAX system.

3. The resultant VAX code will be developed such that it will execute on the Perkin-Elmer system as well.

4. The VAX system will be considered as the resident ARTBASS-M development system.

As in almost any study involving the relocation of existing code from one computer system to another, certain areas of commonality are noticed. To facilitate and reduce the

differences in support code and to enhance system maintainability and configuration control, a list of future options that will further unify the support code on both systems will be noted.

An overview of the findings of this study is presented in the following figure (Figure 1). At the time of this study, the standards and conventions used to implement ARTBASS-M on the PERKIN-ELMER computer system are not known. Certain of the findings of this study may not apply to the actual conversion situation. For example, assembly language may not actually be present on ARTBASS-M, so rewriting that code may not be necessary.

SECTION/SUBJECT	SYNOPSIS OF FINDINGS
FORTRAN:	
Constructs	No major difficulties.
Intrinsic functions	No expected problems.
Data manipulation functions	Underlying code rewritten; minimal problems.
Character manipulation functions	Underlying code rewritten; minimal problems.
Language extensions	User written utility required to format at transfer time.
Input/Output functions	Underlying code rewritten; medium difficulty.
ASSEMBLY LANGUAGE:	All code rewritten; major undertaking; medium difficulty.
INPUT/OUTPUT:	
File management	Nontransferable; machine dependent.
File organization	No expected problems.
EXECUTION:	
Overlays	Not supported on VAX; medium difficulty.
Task and process development	Nontransferable; machine dependent; low difficulty.
Task communication	Most difficult part of conversion; high difficulty.
COMMAND LANGUAGE:	Nontransferable; low difficulty.
CODE UNIFICATION:	New code; new capability; medium-to-high difficulty.

Figure 1. Synopsis of study findings by section.

SECTION/SUBJECT	SYNOPSIS OF FINDINGS
TRANSFER PROCEDURES:	
FORTRAN transfer	Perkin Elmer to VAX medium difficulty; VAX to Perkin Elmer low difficulty.
System utility transfer	New code; medium difficulty.
Scenario data base processing	Single, one-way transfer; medium difficulty.
Input/Output transfer	I/O capability same; low difficulty.
Front-end interface	Interactive system configuration dependent; high difficulty.
Job initiation and control	Nontransferable; low difficulty.
FRONT-END ARCHITECTURE:	Partial transferability; medium difficulty.

Figure 1. Synopsis of study findings by section (continued).

SECTION 2
CONSIDERATIONS FOR THE TRANSFER OF ARTBASS-M CODE

2.1 HARDWARE.

2.1.1 Perkin-Elmer (Model 3240).

The Perkin-Elmer 3240 is a 32-bit machine with a memory capacity of 256,000 bytes to 16 million bytes. The metal-oxide semiconductor (MOS) memory is installed in 256,000 increments with a limit of four million bytes per memory bank.

The input/output devices attached to the computer are classified according to their required speed. The slower peripherals, such as printers, consoles and card readers interface with the computer through the multiplexor bus. The bus can handle up to 1023 devices. The high speed devices, such as disk and tape drives, interface through the Direct Memory Access (DMA) bus. Four DMA buses are available and each bus has eight ports.

The processor has a 8192 byte cache for fast processing of instructions and it has a user Writable Control Store of 8192 bytes for fast execution of commonly used application utilities. A Floating Point Processor is available to increase the processing speed of double-precision and floating point instructions. The processor also provides the capability of having one megabyte of shared memory, which can be shared by up to fourteen Central Processing Units.

The instruction set in the Perkin-Elmer consists of the following major functions:

1. Load/store halfwords, fullwords and multiple words
2. Fixed-Point arithmetic
3. Logical operations (AND, OR, Exclusive OR, Compare and Test)
4. Logical and arithmetic shifts and rotates
5. Bit string and bit manipulation
6. Floating-point arithmetic on single (32-bit) and double (64-bit) precision operands

7. Status and control functions

8. List operations

9. Data handling operations

10. Input/output

11. Byte manipulations

12. Writable Control Store operations

13. Mixed floating-point transfers

14. Privileged system functions

15. Storage-to-storage instructions

16. Decimal conversion instructions

The Perkin-Elmer system provides the following software languages:

1. CAL and CAL MACRO

2. FORTRAN VII

3. RPG II

4. RELIANCE

5. BASIC II

6. COBOL

2.1.2 VAX 11/780.

The Digital Equipment Corporation VAX 11/780 is a 32-bit machine with a physical memory capacity of 256,000 bytes to eight million bytes. The computer has a virtual memory system, which allows a user to address up to four billion bytes of virtual address space. The MOS memory is installed in 256,000 byte increments.

The slower input/output devices, such as card readers, printers and consoles, interface through the UNIBUS. Four UNIBUS adapters can be installed in the computer and up to fifteen devices can be attached to each bus. The high speed input/output devices, such as disk and tape drives, interface with the computer through the MASSBUS. Four MASSBUS adapters can be installed in the computer and up to eight devices can be attached to each bus.

The processor has a 8192 byte cache for fast processing of instructions and it has a user Writable Control store of 24 kilobytes for fast execution of commonly used application utilities. A Floating-Point Accelerator is available to increase the processing speed of double-precision and floating-point instructions. The processor provides for one to eight megabytes of memory that can be shared by more than one Central Processing Unit.

The instruction set on the VAX 11/780 includes the following major functions:

1. Integer and logical instructions
2. Floating-point instructions
3. Packed-decimal instructions
4. Character-string instructions
5. Bit-field instructions
6. Queue manipulation instructions
7. Address manipulation instructions
8. User-programmed general register control instructions
9. Branch, jump and case instructions
10. Subroutine call instructions
11. Procedure call instructions

Digital Equipment Corporation provides the following software languages for the VAX 11/780:

1. MACRO assembler
2. FORTRAN
3. COBOL
4. BASIC
5. C
6. PL/I

7. PASCAL

8. CORAL 66

9. BLISS-32

2.1.3 Hardware Assessment.

The Perkin-Elmer 3240 and the VAX 11/780 are both 32-bit machines with very similar architectures. The Perkin-Elmer has twice the memory capacity of the VAX; however, for this application, eight megabytes of memory is more than sufficient. The VAX is a virtual memory system and the Perkin-Elmer is not; therefore, overlaying of subroutines might be required on the Perkin-Elmer.

Both computers have sufficient buses and ports to attach the required peripherals. The instruction sets of the two computers provide the capabilities necessary to perform intertask communications and bit, byte, floating points instructions required by the math model and front-end. The VAX 11/780 has more languages available than the Perkin-Elmer; however, both machines have FORTRAN and assembler which are required by the math model and front-end.

The literature, provided by both vendors, indicates that the computing power of the Perkin-Elmer 3240 and the VAX 11/780 is approximately equal.

2.2 FORTRAN.

Although FORTRAN is a higher-level language and is designed to be portable between machines, most computer manufacturers have taken some liberties with the implementation of the language. However, most manufacturers market their systems as having met FORTRAN 77 standards (American National Standard Programming Language FORTRAN, ANSI X3.9-1978). The difficulty with meeting FORTRAN 77 standards though, is that it merely specifies the form and establishes the interpretation of programs expressed in the FORTRAN language. It does not dictate how to implement the language.

The scope of FORTRAN 77 is very explicit. It includes the following items:

1. The form of a program written in the language.

2. Rules for interpreting the meaning of a program and it's data.

3. The form of writing input data to be processed by the program.

4. The form of the output data generated by the program.

The FORTRAN 77 standard does not specify the following items:

1. The mechanism by which programs are transformed for use.

2. The method of program and data transcription to or from the data processing medium.

3. The operations required for setup and control of the use of programs on a system.

4. The results when the rules of the standard fail to establish an interpretation.

5. The size or complexity of a program and its data that exceeds the capacity or capability of a system.

6. The range or precision of numeric quantities and the method of result rounding.

7. The physical properties of input/output records, files, and units.

8. The physical properties and implementation of storage.

The FORTRAN 77 standard generally refers to permissible forms and relationships for standard-conforming programs rather than for processors.

This section discusses those properties and conditions of FORTRAN that usually effect the transfer of code between computers. The main areas of consideration will concern FORTRAN constructs, intrinsic functions, data and character manipulation functions, any language extensions, and input/output functions.

2.2.1 Constructs.

Constructs are the building blocks of programs. They define the forms and relationships between well defined program units. They are designed to specify the syntax and interpretation of sets of instructions. This section deals with the FORTRAN 77 constructs adopted by both systems.

2.2.1.1 Perkin-Elmer. The Perkin-Elmer series of 3200 and 3400 machines are "fully compliant with FORTRAN 77". This means that they have met the form and constructs of the FORTRAN 77 standard. They have, however, also developed and implemented "language extensions" which augment the FORTRAN 77 standard in ways which they believe will provide "increased programmer convenience" and enhance the usability of their machines.

2.2.1.2 VAX 11/780. The VAX 11/780 systems also meet the form and constructs of the FORTRAN 77 standard. They have, however, also developed and implemented "language extensions" which augment the FORTRAN 77 standard in ways which they believe will provide increased programmer convenience and enhance the usability of their machines.

2.2.2 Intrinsic Functions.

Intrinsic functions are functions which are available through system libraries at compile time. These functions include:

1. Elementary functions
   - Square root
   - Exponential
   - Trigonometric
   - Logarithm
   - Inverse trigonometric
   - Hyperbolic
2. Min and max functions
3. Type conversion functions
4. Miscellaneous functions
   - Remainder
   - Complex number manipulation
   - Absolute value
   - Sign propagation

- Positive difference
- Truncation and rounding
- Double-precision results from single-precision variables

The particular concern with using intrinsic functions is in three areas. These are:

1. Calling arguments and sequences.
2. Register usage.
3. Error returns
4. Default values.
5. Function name.

Since these intrinsic functions are library routines, their calling sequence and arguments are fixed and occur in a specific order. Further, an intrinsic function on one machine may not have the same number of arguments as on another.

Another potential area of concern is the use of registers as part of the branch and link connections with the intrinsic functions. This is of paramount importance when executing user developed assembler language code and the logic dictates a call to an intrinsic function. It becomes important to have the correct values loaded into the appropriate registers. Furthermore, it is necessary to verify that after the return from the intrinsic function, the values have not been "clobbered" or overwritten.

Error returns and default values are not usually standard on different systems. It is possible that arguments that are within allowable ranges on one system are not within range on another. For example, the call to the tangent intrinsic function, which computes the tangent of an angle given its sine and cosine, is mathematically defined for all angles except for vertical lines (in which case the cosine of the angle is zero).

Some systems will produce a fatal error and processing halts. Other systems will recognize the error but return erroneous values. Still other systems will return a default value of zero. Not all default values returned will necessarily be the same on all machines.

Lastly, intrinsic functions that provide the same capability may not have the same name. In this case, a desired function call on one system will not be the same on another system. Figures 2 through 5 show the intrinsic functions available on both systems.

2.2.2.1 Perkin-Elmer. As shown in Figures 2 through 5, the Perkin-Elmer system provides an extensive assortment of intrinsic functions. They represent commonly required computations and are predefined with respect to the name of the routine and the type of data associated with the arguments.

In some instances, the Perkin-Elmer system allows a group of intrinsic functions with the same mathematical definitions to be given a generic name. When this generic name is used within an expression, the specific function invoked depends upon the type of the actual arguments. The specific function names are the ones shown in Figures 2 through 5.

2.2.2.2 VAX 11/780. Figures 2 through 5 also show the VAX 11/780 intrinsic functions. The VAX also supports generic and specific intrinsic function names. The actual routine that is invoked is resolved at compilation time and is determined by the type of arguments that are used.

The VAX system also provides a few functions that the Perkin-Elmer system does not. However, for the types of applications the math model uses, these functions will not become a part of the operational system.

2.2.3 Data Manipulation Functions.

Data manipulation functions are those routines which are provided as part of the system which manipulates bits, bytes, halfwords, and performs bit string shifts. These types of routines are becoming more prevalent on systems but are not necessarily standard.

Data manipulation functions are designed to operate on subsets of a word. These subsets are dependent on the word structure implemented for the system. Not all systems have the same word structure. Some systems number positions or locations from the left, while others are from the right. In addition, the position or location may be numbered starting at zero or one on a particular system.

Data manipulation functions suffer from the same problems as intrinsic functions. However, user written routines can be developed that always perform data manipulations in the same

DEFINITION	SYMBOLIC NAME	DATA TYPE: ARGUMENT	DATA TYPE: RESULT	REMARK
(a1)**0.5	SQRT	Real	Real	Same
	DSQRT	DP	DP	Same
	CSQRT	Complex	Complex	Same
	CDSQRT	DP cmplx	DP cmplx	Same
e**(a1)	EXP	Real	Real	Same
	DEXP	DP	DP	Same
	CEXP	Complex	Complex	Same
	CDEXP	DP cmplx	DP cmplx	Same
log(a1)	ALOG	Real	Real	Same
	DLOG	DP	DP	Same
	CLOG	Complex	Complex	Same
	CDLOG	DP cmplx	DP cmplx	Same
	ALOG10	Real	Real	Same
	DLOG10	DP	DP	Same
sin(a1)	SIN	Real	Real	Same
	DSIN	DP	DP	Same
	CSIN	Complex	Complex	Same
	CDSIN	DP cmplx	DP cmplx	Same
cos(a1)	COS	Real	Real	Same
	DCOS	DP	DP	Same
	CCOS	Complex	Complex	Same
	CDCOS	DP cmplx	DP cmplx	Same
tan(a1)	TAN	Real	Real	Same
	DTAN	DP	DP	Same
arcsin(a1)	ASIN	Real	Real	Same
	DASIN	DP	DP	Same
	ASIND	Real	Real	Vax
	DASIND	DP	DP	Vax
arccos(a1)	ACOS	Real	Real	Same
	DACOS	DP	DP	Same
	ACOSD	Real	Real	Vax
	DACOSD	DP	DP	Vax

Note: Indicated differences between the two systems are not expected to be used.

Figure 2. Elementary intrinsic functions.

DEFINITION	SYMBOLIC NAME	DATA TYPE: ARGUMENT	DATA TYPE: RESULT	REMARK
arctan(a1)	ATAN	Real	Real	Same
	DATAN	DP	DP	Same
	ATAND	Real	Real	Vax
	DATAND	DP	DP	Vax
arctan(a1/a2)	ATAN2	Real	Real	Same
	DATAN2	DP	DP	Same
	ATAN2D	Real	Real	Vax
	DATAN2D	DP	DP	Vax
sinh(a1)	SINH	Real	Real	Same
	DSINH	DP	DP	Same
cosh(a1)	COSH	Real	Real	Same
	DCOSH	DP	DP	Same
tanh(a1)	TANH	Real	Real	Same
	DTANH	DP	DP	Same

Note: Indicated differences between the two systems are not expected to be used.

Figure 2. Elementary intrinsic functions (Continued).

DEFINITION	SYMBOLIC NAME	DATA TYPE: ARGUMENT	DATA TYPE: RESULT	REMARK
max(a1,...,aN)	AMAX0	Integer	Real	Same
	AMAX1	Real	Real	Same
	DMAX1	DP	DP	Same
	MAX0	Integer	Integer	Same
	MAX1	Real	Integer	Same
min(a1,...,aN)	AMIN0	Integer	Real	Same
	AMIN1	Real	Real	Same
	DMIN1	DP	DP	Same
	MIN0	Integer	Integer	Same
	MIN1	Real	Integer	Same

Note: Indicated differences between the two systems are not expected to be used.

Figure 3. Min and max intrinsic functions.

DEFINITION	SYMBOLIC NAME	DATA TYPE: ARGUMENT	DATA TYPE: RESULT	REMARK
dble(a1)	DBLE	Real	DP	Same
		Complex	DP	P&E
		DP cmplx	Real	P&E
	DFLOAT	Integer	DP	Same
real(a1)	FLOAT	Integer	Real	Same
	SNGL	DP	Real	Same
	REAL	Integer	Real	Same
		Complex	Real	Vax
	DREAL	DP cmplx	DP	Same
int(a1)	IDINT	DP	Integer	Same
		Complex	Integer	P&E
		Integer	Integer	P&E
	IFIX	Real	Integer	Same
	INT	Real	Integer	Same
	INT2	Real	Integer	Same
		DP	Integer	P&E
		Complex	Integer	P&E
	CMPLX	Real	Complex	Same
		Integer	Complex	Same
		DP	Complex	Same
	DCMPLX	Real	DP cmplx	Same
		Integer	DP cmplx	Same
		DP	DP cmplx	Same
		Complex	DP cmplx	Same

Note: Indicated differences between the two systems are not expected to be used.

Figure 4. Type conversion intrinsic functions.

DEFINITION	SYMBOLIC NAME	DATA TYPE: ARGUMENT	DATA TYPE: RESULT	REMARK
Remainder [a1 - (int(a1/a2) *a1)]	MOD	Integer	Integer	Same
	AMOD	Real	Real	Same
	DMOD	DP	DP	Same
Complex number manipulations	AIMAG	Complex	Real	Same
	DIMAG	DP cmplx	DP	Same
	CONJG	Complex	Complex	Same
	DCONJG	DP cmplx	DP cmplx	Same
Absolute value [|a1|]	ABS	Real	Real	Same
	CABS	Complex	Real	Same
	DABS	DP	DP	Same
	CDABS	DP cmplx	DP	Same
	IABS	Integer	Integer	Same
Sign propagation	DSIGN	DP	DP	Same
	ISIGN	Integer	Integer	Same
	SIGN	Real	Real	Same
Positive difference [a1 - min(a1,a2)]	DIM	Real	Real	Same
	IDIM	Integer	Integer	Same
	DDIM	DP	DP	Same
Truncation and rounding	AINT	Real	Real	Same
	DINT	DP	DP	Same
	ANINT	Real	Real	Same
	DNINT	DP	DP	Same
	NINT	Real	Integer	Same
	IDNINT	DP	Integer	Same
DP product of SP variables	DPROD	Real	DP	Same
Zero-extend	ZEXT	Logical	Integer	Vax
		Integer	Integer	Vax

Note: Indicated differences between the two systems are not expected to be used.

Figure 5. Miscellaneous intrinsic functions.

order. This is a relative positive point in that once they are implemented on a system, packing and extracting the data is usually from the same relative locations within the word structure.

Figure 6 summarizes the data manipulation functions that are available on both systems.

2.2.3.1 Perkin-Elmer. The set of standard boolean operations is fully operative on the Perkin-Elmer system. They also offer a standard set of bit processing routines. As a relative plus, Perkin-Elmer also has byte processing capabilities.

The areas wherein Perkin-Elmer does not have the same or equivalent functions is relatively minor. There are routines that can be written to provide the missing support; and, in fact, the routines can be made general enough to run on either machine without recoding when transfers are made between systems.

2.2.3.2 VAX 11/780. As shown in Figure 6, the data manipulation functions that are available on the VAX system are, in general, supported on the Perkin-Elmer system. The primary differences may present major difficulties. These functions are the bit extraction, bit set, bit test, and bit clear.

These four functions are used frequently by the math model since most of its data tables tend to be bit field oriented. Therefore, any use of these VAX system functions will have to be limited since the Perkin-Elmer system has no equivalent. However, as mentioned above, general service routines can be written that are independent of the system on which they are executing.

2.2.4 Character Manipulation Functions.

Character manipulation functions suffer from the same problems that intrinsic functions and data manipulation functions do. Those are mainly calling arguments and sequences. However, character data is also dependent on the word structure implemented on the machine.

There is a standard character set called ASCII. These characters have a fixed binary representation in memory. The recognition of these characters is not as much a problem as is the format of their locations within the word structure.

Figure 7 summarizes the character manipulation functions that are equivalent on the two systems.

DEFINITION	SYMBOLIC NAME	DATA TYPE: ARGUMENT	DATA TYPE: RESULT	REMARK
and(a1,a2)	IAND	Integer	Integer	Same
or(a1,a2)	IOR	Integer	Integer	Same
eor(a1,a2)	IEOR	Integer	Integer	Same
Bitwise complement	NOT BCMPL	Integer Integer	Integer Routine	Same P&E
a1 logically shifted left a2 bits	ISHFT	Integer	Integer	Same
Extract bits a2 through (a2 - a1 - 1) from a1	IBITS	Integer	Integer	Vax
Return value of a1 with bit a2 set	IBSET BSET	Integer Integer	Integer Routine	Vax P&E
Test bit a2 of a1 to determine if it is set	BTEST	Integer	Logical	Same
Return value of a1 with bit a2 clear	IBCLR BCLR	Integer Integer	Integer Routine	Vax P&E
Circularly shift rightmost a3 bits of a1 by a2 places	ISHFTC	Integer	Integer	Vax
Load byte	ILBYTE	Integer	Routine	P&E
Store byte	ISBYTE	Integer	Routine	P&E
Clear byte	ICBYTE	Integer	Routine	P&E
Complement byte	INBYTE	Integer	Routine	P&E

Figure 6. Data manipulation functions.

DEFINITION	SYMBOLIC NAME	DATA TYPE ARGUMENT	DATA TYPE RESULT	REMARK
Returns length of the character expression	LEN	Char	Integer	Same
Returns the position of a2 substring in character expression a1	INDEX	Char	Integer	Same
Returns the character that has a1 ASCII value	CHAR	Logical Integer	Char Char	Same Same
Returns the ASCII value that has a1 character	ICHAR	Char	Integer	Same
Character relationals	LLT LLE LGT LGE	Char Char Char Char	Logical Logical Logical Logical	Same Same Same Same

Figure 7. Character manipulation functions.

2.2.4.1 Perkin-Elmer. Due to the standardization of the ASCII character set, Perkin-Elmer can process the full set of character manipulation functions. The suite of software routines represents the gamut of current processing techniques. Further, its set of capabilities is the same as on virtually every other commerically available system.

2.2.4.2 VAX 11/780. The VAX system has no character manipulation functions that are not standard. In fact, its set of function offerings is exactly the same as the Perkin-Elmer.

2.2.5 Language Extensions.

Although "language extensions" may or may not have a definite meaning on a system, here we shall define language extensions as those capabilities and features which are not an inherent part of the FORTRAN 77 language but are available to provide system management and configuration control, maintain code integrity, and assist the programmer in code development. These language extensions are very powerful and exceedingly helpful, and potentially a major hinderance to the transfer of code between systems.

The major difficulty with language extensions is that they are highly machine dependent and are usually tailor-made to take full advantage of the system architecture. There are very few, if any, ways to automatically convert these features. For general system configuration control and integrity, some degree of language extensions will be used on both systems.

Additionally, some systems have translators which will convert FORTRAN code to FORTRAN 77 code. This is a nice feature but it is not necessarily efficient nor particularly useful. For example, almost all FORTRAN code can be optimized by the system compiler. The resultant production system is then developed to produce correct results with the optimized code.

Processing this production code through a translator would then convert this code into FORTRAN 77 code. On the new system, this converted code might or might not be optimized by the new compiler, but certainly not necessarily in the same manner. The resultant production system then has two potentially fatal hazards. The first is the automatic conversion into FORTRAN 77 code. The second is the new generated object code.

2.2.5.1 Perkin-Elmer. Perkin-Elmer offers several attractive language extensions. These include the following items:

1. File inclusion ("$INCLUDE")
2. In-line assembly code
3. In-line subroutines
4. In-line debug code

Each of these capabilities will be covered in the following paragraphs.

File inclusion is the single, most powerful configuration control device available. It allows the user to insert into the code lines of source code from another file at the spot where the "$INCLUDE" command is placed. This is of prime importance for entities like common blocks and parameters. This ensures that only one version of the code is available and that only the current version is used within the system.

In-line assembly is probably the single most difficult extension to back out and replace. This is so because the assembler code has been written to handle a specific logic sequence and there is no set way to replace it. At times, it may be easier to replace the assembler code with a new subroutine call. Other instances may indicate that elementary intrinsic functions can be used. In-line assembly code will have to be handled on a case-by-case basis.

In-line subroutines will be easy to replace. Instead of loading the subroutine executable code directly into the code, a normal branch can be effected and no system degradation will occur.

In-line debug will, similarly, be easy to replace. On the Perkin-Elmer, the indicator for debug code is "X". Each line of code preceded by this indicator can be eliminated in the new source code. There will be no need, necessarily, to transfer this code.

2.2.5.2 VAX 11/780. The VAX system allows two similar language extensions. They are the following:

1. File inclusion ("INCLUDE")
2. In-line debug code

These will be examined in the following paragraphs.

File inclusion on the VAX allows the user to insert into the code lines of source code from another file at the spot where the "INCLUDE" command is placed. This should be used for entities like common blocks and parameters. This ensures that only one version of the code is available and that only the current version is used within the system. This will help system configuration control.

In-line debug may not really be needed on the VAX code. There are two options available. One is to retain the debug but convert its debug indicator symbol to that recognized by the VAX system. The second is to convert the debug into a "formatted text report" form that is controlled through variables in the data base.

2.2.6 Input/Output Functions.

Input and output functions are, in general, standard on all systems. However, there are always limits to their capabilities. These usually involve the rate of data transfer and the amount of data transfer. The math model tends to be an input/output bound program as well as requiring the writing of thousands of bytes of data every 60 seconds.

The writing of massive quantities of data requires a fine balance between the number of I/O requests and the amount of data that can be physically transferred. In this respect, a single I/O transfer request is more efficient and quicker than several requests. However, there is a limit to the amount of data that can be transferred.

Custom written code which will transfer the maximum amount of data per call is the usual solution. However, this code needs to be rewritten on the new host machine to take advantage of its architecture.

2.2.6.1 Perkin-Elmer. The Perkin-Elmer system allows data to be written in groups of 512 byte records. However, the total number of groups written at any one time is less than the actual number of bytes to be written by the model. It is assumed that utility input and output routines have been developed for use on the Perkin-Elmer system.

These utility routines can be used as the basis of similar routines written for the VAX system. It will be fairly straightforward to do this and locate the appropriate places for their calls.

2.2.6.2 VAX 11/780. The VAX system has its files organized around 512 byte blocks. It also only allows the writing of approximately 32,000 bytes at a time as a maximum. To allow the transfer of larger amounts of data, user written and

developed code can be generated that will automatically handle these larger amounts of data.

2.2.7 Assessment.

The purpose of this section is to illuminate discrepancies between the Perkin-Elmer and VAX 11/780 machines and to indicate the recommended solution to those discrepancies.

2.2.7.1 Constructs. In general, the constructs of FORTRAN 77 are the same on both machines. Mild variations are expected but should cause no major difficulty. Any differences and anomalies will be addressed on a case-by-case basis.

2.2.7.2 Intrinsic Functions. As shown in Figures 2 through 5, the intrinsic functions available on both systems are the same. This means that the calling arguments, default values, and error conditions are the same. There are no expected problems with intrinsic functions.

2.2.7.3 Data Manipulation Functions. Figure 6 summarizes the data manipulation functions. There is a wide discrepancy among the types of routines and their capabilities. However, the symbolic names used on the Perkin-Elmer system will be retained on the VAX system. Only the underlying code will be altered on the VAX. Further, this code will not be transferred back to the Perkin-Elmer system since it already exists and functions correctly there.

2.2.7.4 Character Manipulation Functions. These functions should prove to be among the easiest codes to be converted. These functions should present minimal, if any, problems.

2.2.7.5 Language Extensions. Both systems have functionally similar but different implementation of these functions. A commonality will need to be decided on and a utility routine implemented to accommodate format differences.

2.2.7.6 Input/Output Functions. The majority of input/output capabilities are the same. However, for certain specific areas, a new code will need to be written that will be tailored for each machine. This will necessitate two versions of the same code running on the two machines, but they should not need to be altered once they are finally operational.

2.3 ASSEMBLY/MACHINE LANGUAGE.

Assembly language is, by definition, code which is written in machine language. It is, therefore, machine dependent and not transferable.

2.3.1 Perkin-Elmer.

The Perkin-Elmer system allows for users to write and develop assembly language code. It is foolish to expect that no assembly language code has been written. In reality, it is a question of the function, how much, and the complexity of the code that was implemented that will dictate the ease of the transfer.

The Perkin-Elmer system also allows the user to include in-line assembly code into a FORTRAN program. These sections of code will be difficult to replace. In general, these sections of code will be replaced with the appropriate data manipulation routines (see Section 1.2.3). However, this will not always be possible, and for these instances the logic will be replaced with subroutine calls.

2.3.2 VAX 11/780.

The VAX 11/780 system also allows for users to write and develop assembly language code. Any required assembly language code can be developed and made to perform the same function as it did on the other system.

2.3.3 Assessment.

Assembly language code is nontransferable. However, once its function can be ascertained, new code can be written to replace it on the VAX. In addition, it may become evident that a new FORTRAN code, generalized for both systems, could be implemented to reduce the amount of assembly language code. Further, any assembly language code should be placed into libraries resident on the specific machine and not be considered as part of the transfer.

2.4 INPUT/OUTPUT.

Input statements provide the means of transferring data from external media to internal storage or from an internal file to internal storage. Output statements provide the means of transferring data from internal storage to external media or from internal storage to an internal file.

Many I/O statements have a list of entities called an I/O list that follows the list of the I/O specifier. In input statements, these entities become defined with values read from records. In output statements, the values of these entities are written in records.

In addition to the statements that transfer data, there are auxillary input/output statements to manipulate the external medium, or to inquire about or describe the properties of the connection to the external medium.

All of these input/output conventions are covered by FORTRAN 77.

2.4.1 File Management.

File management concerns itself with the maintenance, updating, and general management of all files. This is an operating system function and not a programming consideration. As such, it is independent of FORTRAN 77 standardization and is not transferable.

2.4.1.1 Perkin-Elmer. On the Perkin-Elmer system, input statements can read data from external files, devices, internal files, or buffers and transfer it to internal storage. Output statements write data from internal storage to external files, devices, internal files, or buffers.

The Perkin-Elmer operating system supports both indexed and contiguous file types. Accessing these file types can be performed by direct, random, or sequential methods. In addition, temporary or scratch files can be used and will be deleted immediately upon completion of the program which created them.

The Perkin-Elmer file manager system has a standard set of protocol for the management of permanent files. These include the creation, assignment, and deletion of any identified file. These are all standard capabilities for file management.

The Perkin-Elmer system fully supports the FORTRAN 77 standard.

2.4.1.2 VAX 11/780. On the VAX system, input and output statements translate data from internal (binary) form to external (readable character) form, or vice versa. The VAX operating system further supports sequential, direct access, indexed, and internal I/O.

The VAX file management system also has a standard set of protocol for the management of permanent files. These include the creation, assignment, and deletion of any identified file. These are all standard capabilities for file management.

2.4.2 File Organization.

A file is a collection of logically related records that are arranged in a specific order and treated as a unit. The arrangement or organization of a file is determined, usually, when the file is created. There are three types of file organization. They are sequential, relative, and indexed.

Records in a sequential file are ordered in physical sequence. Each record, except the first, has another record preceding it. Each record, except the last, has another record following it. The physical order in which the records appears is usually identical to the order in which the records were originally written to the file.

Records in a relative file consist of a specified number of fixed-length cells ordered in physical sequence. These cells are numbered from 1 (the first) to N (the last). Each number represents the location of a record relative to the beginning of the file. Each cell either contains a single record or is empty. The cell (record) number is used to refer to specific records in the file.

Records in an indexed file are ordered by fields in the records called keys. A key is a data field in the record of an indexed file. When the indexed file is created, a field within the file's record is determined to be the key. The contents of these fields are then used to identify specific records for subsequent processing. The length of a field key, and it's relative position in the record, are identical for all records in the file.

There is at least one key for an indexed file. This mandatory key is called the primary key of the file, and has a unique value for each record. Other keys may be defined and are called alternate keys. An alternate key consists of a field that is held in common by, and located in the same position in, each record in the file. Both primary and alternate keys may be used to identify a record for retrieval. An alternate key does not need to have a unique value in each record.

Access mode is the method a program uses to retrieve and store records in a file. The access mode is specified as part of each I/O statement.

File organization is directly linked to auxiliary I/O statements. These statements open and close files, specify the attributes of the file, determine or change the way a file or unit is assigned, reposition a file to a previous record, or write endfile records.

Figure 8 shows the various types of FORTRAN 77 I/O statements and Figure 9 shows access modes for each file organization on the two systems.

2.4.2.1 Perkin-Elmer. As shown by Figure 8, the Perkin-Elmer system supports the FORTRAN 77 specification as far as file organizations are concerned.

FORTRAN 77 STATEMENT NAME	STATEMENT CATEGORY											
	SEQUENTIAL			DIRECT			INDEXED			INTERNAL		
	F	L	U	F	L	U	F	L	U	F	L	U
Read	X	X	X	X	-	X	X	-	X	X	-	-
Write	X	X	X	X	-	X	X	-	X	X	-	-
Rewrite	-	-	-	-	-	-	X	-	X	-	-	-
Accept	X	X	-	-	-	-	-	-	-	-	-	-
Type	X	X	-	-	-	-	-	-	-	-	-	-
Print	X	X	-	-	-	-	-	-	-	-	-	-

Notes: 1. "F" denotes formatted.
2. "L" denotes list-directed.
3. "U" denotes unformatted.

Figure 8. Available I/O statements.

FILE ORGANIZATION	ACCESS MODE		
	SEQUENTIAL	DIRECT	KEYED
Sequential	Yes	Yes (1)	No
Relative	Yes	Yes	No
Indexed	Yes	No	Yes

Notes: 1. Records must be fixed-length.

Figure 9. Access modes for each file organization.

2.4.2.2 VAX 11/780. As shown by Figure 9, the VAX system also supports the FORTRAN 77 specification as far as file organizations are concerned.

2.4.3 Assessment.

FORTRAN 77 file management and organization provides a structured manner in which to manipulate data files. Both systems fully support the file characteristics specified by FORTRAN 77. There should be no difficulty in transferring this type of FORTRAN code. In fact, there should be no changes at all to any file manipulation FORTRAN code in the transfer of the ARTBASS code.

2.5 EXECUTION.

An executable program is a collection of program units that consist of exactly one main program and any number, including none, of subprograms and external procedures. The running of the executable program is called execution.

During execution, executable statements in the program units are implemented and executed in the order in which they appear. Execution of an executable program begins with the execution of the first executable statement of the main program. When an external procedure specified in a program unit is referenced, execution begins with the first executable statement that follows the FUNCTION, SUBROUTINE, or ENTRY statement that specifies the referenced procedure as the name of a procedure.

This section discusses the implementation of program execution on the two systems. In particular, overlays, tasks and processes, and task communication are discussed in the light of how they facilitate the program execution.

2.5.1 Perkin-Elmer.

2.5.1.1 Overlays. The Perkin-Elmer system provides a means to execute a program in an area of main memory that is not actually large enough to contain the entire task at one time. The program linker is used to divide such a program into nodes, a collection of modules and common blocks, which are loaded as needed. Only one node, the root, must remain in main memory throughout the execution of the program. The other nodes reside on, and are fetched from, disk when needed.

To ensure the integrity of the overlayed program, an overlay structure must be carefully designed. This structure is a tree structure that shows which nodes of a program occupy

the same main memory at different times. The main routine must reside in the root node throughout the execution of the task.

The Perkin-Elmer OVERLAY command specifies the start of a node and the node's relative position within the tree structure. In addition, any run-time library files can be specified so that remaining entry points can be resolved.

Each node has a fixed length in bytes. The total size of a task depends on both the routine composition of each node and the structure of the overlay tree. An overlay structure can be represented by a set of parallel paths. A path can be defined as a particular set of nodes with one at each level, and each of which is a descendent from the previous level.

Therefore, the total size of a task is determined by the path whose node size adds up to the greatest number of bytes. Normally, by using the cross reference map from the linker, a manually created call-tree representation of a program can be generated as an aid in determining the smallest possible task size.

Normally, the placement of a common block or global block within an overlayed task is determined by where the block is referenced. Blank common is always positioned in the root. Named common and global blocks are initially positioned by the linker no closer to the root than any particular reference to the block.

There are two consequences to this positioning scheme. The first is that named common and global entities are initialized every time the overlay is fetched from disk. The second consequence is that two or more overlays can have their own separate and private copies of a common or global entity. These copies could then contain different values.

In addition to common blocks and global entities, implicitly saved local entities are also affected by overlaying a program. Suppose a program containing an implicitly saved local entity depends on the value of that entity to remain unchanged between invocations. The value of that entity is well defined at one point during the program execution, but becomes undefined at another.

2.5.1.2 Task and Process Development. The normal program development procedure is divided into three sections, each representing the three processes required to develop a program. These three processes are COMPILE, LINK, and EXECUTE.

COMPILE inputs the source code file to the compiler, starts the compilation, outputs a source listing, checks for compilation errors, and outputs the resultant object code if no errors have occurred. LINK converts the object code produced by the COMPILE process into a task image. It also outputs a link map, checks for link errors, and outputs the executable image to the task image file if no errors have occurred. The EXECUTE process loads the task image, executes the task and outputs the task results.

2.5.1.3 Task Communication. At this time, the form of the intertask communications used for the ARTBASS-M code on the Perkin-Elmer system is not known.

2.5.2 VAX 11/780.

2.5.2.1 Overlays. The VAX 11/780 uses the Virtual Memory System (VMS) operating system. The VMS system is a virtual memory management system and as such it has no program overlay capabilities.

2.5.2.2 Task and Process Development. The VAX normal program development procedure is divided into four sections, each representing the processes required to develop a program. These four processes are EDIT, FORTRAN, LINK, and RUN.

The process of EDIT is the editing of the source code which is resident in a program file. This is what the programmer does to create and alter the source program in order to make it operate correctly.

Following the editing of the source code, the program is compiled. The FORTRAN command inputs the source code file to the compiler, starts the compilation, outputs a source listing, checks for compilation errors, and outputs the resultant object code if no errors have occurred.

Next, the object code(s) need to be linked to form the task image. LINK converts the object code produced by the FORTRAN process into a task image. It also outputs a link map, checks for link errors, and outputs the executable image to the task image file if no errors have occurred.

The RUN process loads the task image, executes the task and outputs the task results.

2.5.2.3 Task Communication. The VAX system offers several features to facilitate the communication interfaces between tasks. These features can also be used in conjunction with each other. Figure 10 lists the features available.

AVAILABLE FEATURE	DESCRIPTION OF MAIN USE
Common event flags	Notify process of event completion; synchronize access to a resource.
Mailboxes	Pass messages or other data between processes.
AST service routines	Execute desired routine in response to an external event, regardless of when the event occurs.
Hibernation and suspension	Activate subprocesses and detached processes only when they are needed.
Global sections	Share data or code.
Sharable images	Share data or code.

Figure 10. VAX communication, synchronization, and sharing features.

The difficulty with using these features as provided is that they are also used by the VAX system. Consequently, certain ranges of features are not available for user application use since they are reserved for system communications, synchronization, and sharing. Therefore, the full gamut is not really available.

2.5.3 Assessment.

2.5.3.1 Overlays. Overlay structures, if used on the Perkin-Elmer system, will not be a problem to unravel. On the VAX system, overlaying is not supported. All appropriate common blocks and data entities will be located either in the main program or at an appropriate level determined by the linker.

The only potential difficulty will be in the restoration of the VAX data structures onto the Perkin-Elmer system. This can be minimized by the use of linker commands that force the location of common blocks in relationship to main memory.

At this time, the use of overlays for the ARTBASS-M code on the Perkin-Elmer sytem is not known. However, if no overlays are used, then the problems mentioned here become a moot point.

2.5.3.2 Task and Process Development. Both the Perkin-Elmer and VAX systems allow the same types of developmental processes. What should be developed is a command file that will automatically compile all of the required source code program files, and then link them together to create the executable image. As an option, this automatic command file might also begin the task execution.

If the automatic command file does not have the task execution command, then one should be available as a stand-alone command. Figures 11 and 12 show representative automatic task creation command files.

2.5.3.3 Task Communication. Real-time implications often consist of related programs running as several processes. These processes may be detached processes, or detached processes with one or more subprocesses. These processes usually need to communicate with each other and share common data or code.

The symbolic names used by the ARTBASS-M code will be retained on the VAX. The code will, however, be altered on the VAX to conform to VAX system architecture. This new code will not be transferred back to the Perkin-Elmer system since it already exists on that system and it executes correctly. This function will probably be the most difficult to convert on the VAX.

PROCESS DESCRIPTION	OPERATING SYSTEM COMMANDS
Load the compiler	LOAD F7D,30 ASSIGN 1,@1.FTN XALLOCATE @1.OBJ,IN,126/2 ASSIGN 2,@1.OBJ XALLOCATE @1.LST,IN,132/2 ASSIGN 3,@1.LST
Compile source	START ,@2 $IFG 1 $WRITE COMPILATION ERRORS $CLEAR $ENDC
Delete old executable	XDELETE @1.TSK XALLOCATE @1.MAP,IN,132/2
Build linker commands	$BUILD LINK.CMD ESTABLISH TASK MAP @1.MAP,ALPHABETIC, ADDRESS,XREF OPTION FLOAT,DFLOAT, WORK=(C00,C00), SYSSPACE=FFFF INCLUDE @1.OBJ SHARED GLOBAL LIBRARY F7RTL.OBJ/S BUILD @1.TSK END $ENDB
Load linker Link the object code	LOAD MTM:LINK/S,20 START,COMMAND=LINK.CMD, LOG=CON: $IFNE 0 $WRITE LINK ERRORS $CLEAR $ENDC
Load the executable	LOAD @1.TSK ASSIGN 1,file
Assign all files	. . .
Run task image	ASSIGN n,file START

Figure 11. Perkin-Elmer task and process development command file.

PROCESS DESCRIPTION	OPERATING SYSTEM COMMANDS
	$OPEN/WRITE NOUT SYS$OUTPUT
Open input file	$OPEN/READ FILES 'P1'
Read source code file name	$GET: READ/END_OF_FILE=DONE
	FILES NAME
	$WRITE NOUT NAME
Delete old object	$DEL O:'NAME'.OBJ;*
Delete old listing	$DEL L:'NAME'.LIS;*
Compile source	$FORTRAN S:'NAME' -
	/OBJECT=O:'NAME' -
	/CONTINUATIONS=75
	$GOTO GET
	$DONE: CLOSE FILES
	$CLOSE NOUT
Delete old executable	$DELETE O:MODEL.EXE
Delete old link map	$DELETE O:MODEL.MAP
Link	$LINK/FULL/MAP=O:MODEL -
	/EXECUTABLE=O:MODEL -
	S:MODEL/OPTIONS
Run task	$RUN O:MODEL.EXE

Figure 12. Vax task and process development command file.

2.6 COMMAND LANGUAGE.

The system command language is a set of commands that provide the following functions:

1. Interactive program development.
2. Device and data file management.
3. Interactive and batch program execution and control.

These functions are intended for all users of a system, including application programmers, system programmers, operators, and managers.

2.6.1 Perkin-Elmer.

The Perkin-Elmer system provides a set of command language commands. The most important of these commands is the login command which allows the user to access the system. This involves some type of user name and a password. The system then validates that the user is authorized to use the system.

The Perkin-Elmer system then provides an operating environment once it is ready to accept commands. This environment has various characteristics associated with it, among which are the following:

1. An account number,
2. A user identification code,
3. A default disk device and directory name,
4. Default devices for input, output, and error streams,
5. A set of privileges and resource quotas, and
6. A command interpreter.

These characteristics are unique to each user.

Commands consist of English language words that describe what the system is to do. Commands can optionally be modified.

Using these commands, the user can create, access, and update data files and programs. The Perkin-Elmer system

provides the access and control capabilities that are called for by the commands.

The Perkin-Elmer operating system provides concurrent time-sharing multiprogramming and batch job processing. As part of its programming environment, the Perkin-Elmer system provides the following:

1. Commands for program development,
2. Debugging programs,
3. Traceback information, and
4. Exit and condition handlers.

2.6.2 VAX 11/780.

The VAX system provides an extensive set of DCL (DIGITAL Command Language) commands. The most important of these commands is the login command which allows the user to access the system. This involves a user name and a password. The system then validates that the user is authorized to use the system.

The VAX system provides an operating environment when it is ready to accept commands. This environment has various associated characteristics. Among which are the following:

1. An account number,
2. A user identification code,
3. A default disk device and directory name,
4. Default devices for input, output, and error streams,
5. A set of privileges and resource quotas, and
6. A command interpreter.

These characteristics are unique to each user.

Commands consist of English language words (generally verbs) that describe what the system is to do. Commands can optionally contain qualifiers and parameters. Qualifiers modify a command and provide additional information on how to execute the command. Parameters describe the object of the command. In addition, commands may be placed into files and the entire file interpreted as a single command.

Using DCL commands, the user can create, access, and update data files and programs. The VAX Record Management Services (RMS) provide the access and control capabilities that are called for by the DCL commands. Further, files can be defined and accessed from within programs by using RMS or the input/output services of the VAX/VMS operating system directly.

The VAX operating system provides concurrent time-sharing multiprogramming and batch job processing. As part of its programming environment, the VAX system provides the following:

1. Commands for program development;
   - $ EDIT
   - $ FORTRAN
   - $ MACRO
   - $ LINK
   - $ RUN

2. Debugging programs;
   - Local symbol table information,
   - Global symbol information, and
   - Traceback information.

3. Exit and condition handlers.

2.6.3 Assessment.

No assessment can really be made here. Command language is a system dependent capability. Since both systems are interactive systems, they have the same features. What differs is the implementation of those same features and capabilities. Since neither command language can be transferred to the other system, all that can be said for either system is that their command language is adequate for the type of work to be performed.

2.7 CODE UNIFICATION.

After the ARTBASS-M becomes operational on the VAX system and regular development commences, it is recommended that

new, general purpose service routines be developed. These routines should be in the following functional areas:

1. Data manipulation
2. Character manipulation
3. Task communication

There are several reasons for this code unification. First, it will provide a commonality of source code on both systems. Second, the maintainability of the code will increase since the programs will be the same on both systems, thus eliminating the burden of the same function being performed by two different sets of code. Third, the integrity of the code is ensured as much as possible. Fourth, system configuration control is centralized onto one system.

The following sections outline the functions and capabilities of these general service routines.

2.7.1 Data Manipulation Functions.

It is recommended that instead of using the system routines provided, the following set of general purpose routines be implemented:

1. Clear bit
2. Set bit
3. Test bit
4. Put field
5. Get field
6. Address retrieval

These will be explained in the following paragraphs.

The clear bit routine ("CALL CLRBIT(a1,a2)") will clear the a1-th bit in source a2. The set bit routine ("CALL SETBIT(a1,a2)") will set the a1-th bit in source a2. The test bit routine ("TSTBIT(a1,a2)") will return the value "false" if the a1-th bit of source a2 is not set, and the value "true" if it is set.

Any requirements for bit, byte, or halfword placement or extraction as well as bit shifts should be replaced with the general purpose routines to get or put a field. The

rationale for this is to provide more flexibility between the two systems, better data packing management schemes, as well as providing, potentially, a faster algorithm.

The put field routine ("CALL PUTFLD(a1,a2,a3)") will place a1 bits starting at bit position a2 in source a3. The get field routine will consist of two versions, a signed and unsigned version. The signed get field routine ("IGTFLS(a1,a2,a3)") will extract a1 bits starting at bit position a2 in source a3, and it will sign extend the result. The unsigned get field routine ("IGTFLU(a1,a2,a3)") will extract a1 bits starting at bit position a2 in the source a3 and zero extend the value.

The address retrieval routine ("IADRES(a1)") will return the starting address in memory of a1.

All of these routines will be written in assembly language. There will be a single set of routines for the VAX and a corresponding set for the Perkin-Elmer system. All routines will, therefore, be working with data word formats constructed as usual on the respective system. All returned values will also be appropriately justified for the particular system. They would all reside in libraries which would not be transferred between the two systems.

2.7.2 Character Manipulation Functions.

A standard set of routines should be provided to work on both machines without degradation of performance. The following types of routines will need to be provided:

1. Copy character data

2. Binary-to-ASCII data conversion

Character data should be transferred between words by use of the move character function. This function ("CALL MOVE(a1,a2,a3,a4,a5)") will copy a1 characters starting at position a2 in string a3 into position a4 of string a5.

The binary-to-ASCII conversion routine ("CALL BINASC(a1,a2,a3)") should convert to a decimal equivalent the binary integer a1 into an ASCII character string a2 characters long. If the logical value a3 is set to "true", the character string will include leading zeroes. If set to "false", leading zeroes will be suppressed.

2.7.3 Task Communication.

Real-time applications often consist of related programs running as several processes. These processes may be

detached processes, or detached processes with one or more subprocesses. These processes usually need to communicate with each other and to share common data or code.

Interprocess communication often consists of event notification, although it can also involve transmission of messages or other data. Processes within an application can synchronize their operations through effective communications. Processes can also share code or data to reduce the application's physical memory requirements.

Since neither system provides the type of intertask communications that is really an industry recognized standard, an acceptable scheme should be generated. In addition, a readily transportable scheme is required.

The Semaphore Utility routines should provide a set of task synchronization primitives. This will synchronize a global resource that is shared between two asynchronous tasks, since problems can occur if both tasks try to access (one to read and one to write, or both to write) the same global area.

To alleviate this problem, the two tasks can consider the global area as a nonsharable resource. Whenever a task wants to access the area, the task requests its exclusive use. When this exclusive use is granted by the operating system, the area can be processed as desired.

When the task is finished with the area, it signals the operating system that it is through with the resource so that another task may use it. This type of mechanism uses one preassigned global event flag for each resource.

An interprocess queuing system should also be provided. This set of routines will maintain the interprocess queues. It allows the user to place an item on one of the queues, read it independently of its position on the queue, get the queued item, and remove the item from the queue. This system will also maintain a map of allocated blocks in secondary storage associated with the queue file.

SECTION 3
ARTBASS-M CODE TRANSFER PROCEDURES

This section deals with the physical transfer of ARTBASS-M code from the Perkin-Elmer to VAX and vice versa. There are two basic ways to perform this task. One way is a hardware lash-up between the two systems. The second is to transfer code by means of magnetic tape.

The hardware approach would entail the connecting of a cable between the systems. Although feasible, this approach is not advantageous. There are several reasons for not using this approach.

The first reason concerns itself with the frequency of code transfers. In theory, there will be only one transfer of code from the Perkin-Elmer system to the VAX system. Code transfer from the VAX to the Perkin-Elmer system will not necessarily be done on a frequent basis. It would be expected that any VAX to Perkin-Elmer transfer will be accomplished every three to four months. This is not frequent enough to justify a hard cable connection between the systems.

The second reason concerns itself with the speed and volume of data to be transferred. It is expected that approximately 35,000 lines of code will be transferred between the systems roughly four times a year. It will take an hour or less to transfer all model code. This level of speed and volume is not enough to justify a cable connection between the systems.

The third reason concerns itself with the cost, installation, and maintenance of the connection. Disregarding cost, which is an unknown quantity, the installation and maintenance of the connection is potentially a very time consuming task as well as being costly in the long run. This also does not justify a hardware connection between the systems.

Based on the above considerations, it is recommended that all data transfers between the Perkin-Elmer and VAX systems will be accomplished by tape. The remainder of this section will discuss the actual mechanics of the transfer between the systems.

3.1 FORTRAN TRANSFER.

3.1.1 Perkin-Elmer to VAX 11/780.

The initial transfer of Perkin-Elmer resident FORTRAN code to the VAX will include all math model code, support code, and all necessary files. In addition, compilation listings, link load maps, and any necessary support listings will also be made available for the conversion work to be accomplished on the VAX.

The transfer to tape will use the Perkin-Elmer "COPY32" system utility. A VAX compatible blocksize will be specified. Each record of source code will be 80 bytes in length. The file blocksize and record length will, of course, be adjusted appropriately for different types of files other than source code files.

3.1.2 VAX 11/780 to Perkin-Elmer.

Examination of the Perkin-Elmer FORTRAN VII User Guide manual and the FORTRAN VII Reference Manual indicates that the VAX FORTRAN 77 is basically a subset of the FORTRAN 77 used by the Perkin-Elmer system. This will greatly facilitate the VAX developed ARTBASS-M code for its implementation on the Perkin-Elmer system.

As mentioned above in Section 1.2, the maximum amount of VAX system capabilities should be utilized. This will include the use of the "INCLUDE" and "PARAMETER" features. Since the Perkin-Elmer system does not support these features as implemented on the VAX, a program will be developed which will convert the VAX "INCLUDE" or "PARAMETER" format into the Perkin-Elmer format before writing the source to the transfer tape.

The source tape thus created on the VAX system will have the complete source code as well as the appropriate support features of "INCLUDE" and "PARAMETER". A program will also be written for the Perkin-Elmer system that will read the tape and place the source code into its respective files.

3.2 ASSEMBLY LANGUAGE TRANSFER.

3.2.1 Perkin-Elmer to VAX 11/780.

Because assembly language is system dependent, there will be no physical transfer of this code. Instead, assembler code listings will be used to determine the function of the code. This function will then be installed on the VAX. In some instances, this code will be replaced by FORTRAN subroutines to facilitate it's use on the two systems. In other instances, new assembler code will be written to generalize the function for the two systems.

3.2.2 VAX 11/780 to Perkin-Elmer.

As mentioned above, assembler code will not be "transferred" between the two systems. Functionally identical code will be developed, however, to perform the same logic.

3.3 SYSTEM UTILITY TRANSFER.

3.3.1 Perkin-Elmer to VAX 11/780.

In general, wherever possible, system utilities will be used to create tapes of source code and support files. These system utility created tapes will be used to transport all necessary entities to the VAX system.

As mentioned above in Section 3.1, some tailor-made utility programs will be written to minimize the system feature differences. These utility programs will be kept to a minimum and be as general and simple as possible.

3.3.2 VAX 11/780 to Perkin-Elmer.

In general, special utility programs will be used to reformat the VAX code so that it can be processed directly by the Perkin-Elmer system. Further discussions of this are in Sections 3.2 and 3.1 above.

3.4 SCENARIO DATA BASE PROCESSING.

3.4.1 Perkin-Elmer to VAX 11/780.

The transporting of data bases from the Perkin-Elmer system will consist of copying the data to a tape. The copy process will be performed by system utilities, without formats.

Any programs that read data bases will be examined. If the reads are by formats, the code can be transferred with no changes. Unformatted data base reads, however, will need special consideration. Where possible, general purpose binary data read routines will be used.

3.4.2 VAX 11/780 to Perkin-Elmer.

Data base processing routines will be used on both systems. These will be written in FORTRAN except where assembler language is required to interface with system routines. The assembler code will not be transferred, however, it's functional equivalent will exist on the Perkin-Elmer system.

3.5 INPUT/OUTPUT TRANSFER.

3.5.1 Perkin-Elmer to VAX 11/780.

ANSI standard FORTRAN I/O processing will be transferred directly from system to system. This includes READ, WRITE, ENCODE, DECODE, PRINT, and TYPE statements. I/O that deals with binary data transfer will be examined and transferred where possible. Assembler language level I/O will not be transferred.

3.5.2 VAX 11/780 to Perkin-Elmer.

As far as possible, I/O processes will be retained and remain the same on the two systems. However, if a process can be generalized and meet the ANSI FORTRAN standard, the old routine will be altered to make it more general. Assembler language level I/O will not be transferred.

3.6 FRONT-END INTERFACE.

3.6.1 Perkin-Elmer to VAX 11/780.

The actual front-end interface is dependent on the configuration of the interactive system. This will dictate the format and contents of the shared memory data structures.

Certain front-end interfaces will be invariant between the two systems. These will be the event queuing system and the event clusters. These will be handled as will the I/O code. Namely, FORTRAN based code will be transferred as unchanged as possible. Assembly level code will not be transferred.

3.6.2 VAX 11/780 to Perkin-Elmer.

Task communication, synchronization, and shared resources will be standard on the two systems. Subroutine calls will remain the same, but the underlying code may be different to accommodate the particular system.

Since task communication, synchronization, and shared resources are basically state-of-the-art, these functions will remain unaltered. The implementation will most likely be different on the two systems.

3.7 JOB INITIATION AND CONTROL.

3.7.1 Perkin-Elmer to VAX 11/780.

Job initiation and control is a system dependent feature. There will be no transfer of this type of code. However, as much as possible, command language files will be developed that will automatically process files to be transferred to the VAX.

For transferring back to the Perkin-Elmer system from the VAX, a command file will be created that will automatically read the VAX created tape, copy the programs to their respective files, compile all of the programs, link the resultant object code, and execute the ARTBASS-M math model.

3.7.2 VAX 11/780 to Perkin-Elmer.

As much as possible, command language files will be developed that will automatically process files to be transferred to the Perkin-Elmer system. In addition, automatic files will be created for compiling, linking, and executing the VAX ARTBASS-M code.

SECTION 4
FRONT-END ARCHITECTURE

4.1 INTRODUCTION.

The Front-End Architecture includes all hardware and software modules that are part of the man-machine interface.

This section describes the man-machine interface for Combined Arms Tactical Training System (CATTS), ARTBASS-M, NTC Test Support Driver and Mace. Each system's hardware and major software modules are described.

4.2 HARDWARE.

The hardware description of each of the above systems includes the host computers, color graphics devices, bit pads, and alphanumeric terminals and printers.

4.2.1 CATTS.

CATTS is installed on a SIGMA 9 computer with 128k 32-bit words of main memory, 3 disk drives, printer and 3 tape drives. The computer supports three control stations: threat, maneuver, and fire support. The hardware to support the three control stations includes:

1. 1 SIGMA 9 computer
2. 1 Ramtek GX-100 color graphics processor
3. 3 Color cameras and map boards
4. 4 Super Bee alphanumeric terminal
5. 1 Audio recorder
6. 1 Simulated RATT (teletype)
7. 3 Graph tablets and pens
8. 3 Control panels
9. 4 Conrac 19" color monitors
10. 3 TI printers
11. 1 Large screen display

The interaction of the computer hardware and the CATTS model is shown in Figure 13.

4.2.2 ARTBASS-M.

ARTBASS-M is installed on two PERKIN-ELMER 3240s with shared memory and one PERKIN-ELMER 3220. One of the 3240s runs the math model, the other 3240 handles map and graphic displays and the 3220 handles I/O for the graph tablet and the touch sensitive keyboard. ARTBASS-M supports five control stations: two maneuver, one threat, one fire support and one admin/log. The system includes the following hardware:

1. 2 PERKIN-ELMER 3240 computers with shared memory
2. 1 PERKIN-ELMER 3220 computer
3. 5 Lexidata 3400 color graphics processors
4. 8 Lexidata 19" color monitors
5. 5 Multifunction keyboards
6. 5 graph tablets and pens
7. 7 Alphanumeric terminals
8. 7 Control station printers

The interaction of the computer hardware and the ARTBASS-M model is shown in Figure 14.

4.2.3 NTC Test Support Driver.

NTC Test Support Driver is installed on a Digital VAX 11/780. The math model can be interacted with from as many control stations as exist in the NTC system. The hardware at each control station is controlled by the LSI computer in the De Anza color graphics processor. The following hardware is required for each control station:

1. 1 De Anza VC23 color graphics processor
2. 2 VT-105 alphanumeric/graphic terminals
3. 1 Graph tablet and pen
4. 1 Control station printer
5. 1 Large screen display

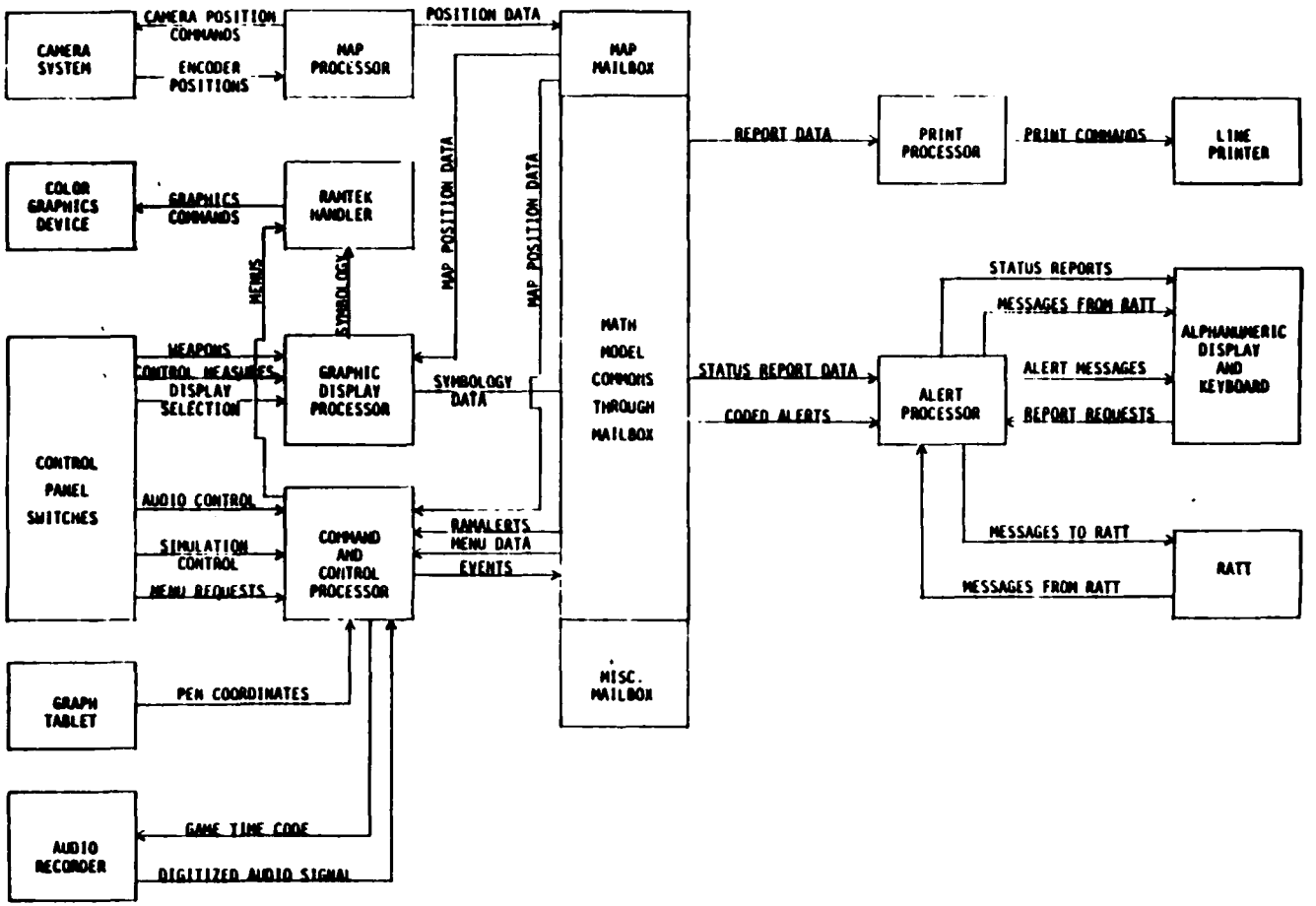


Figure 13. CATTS equipment connection.

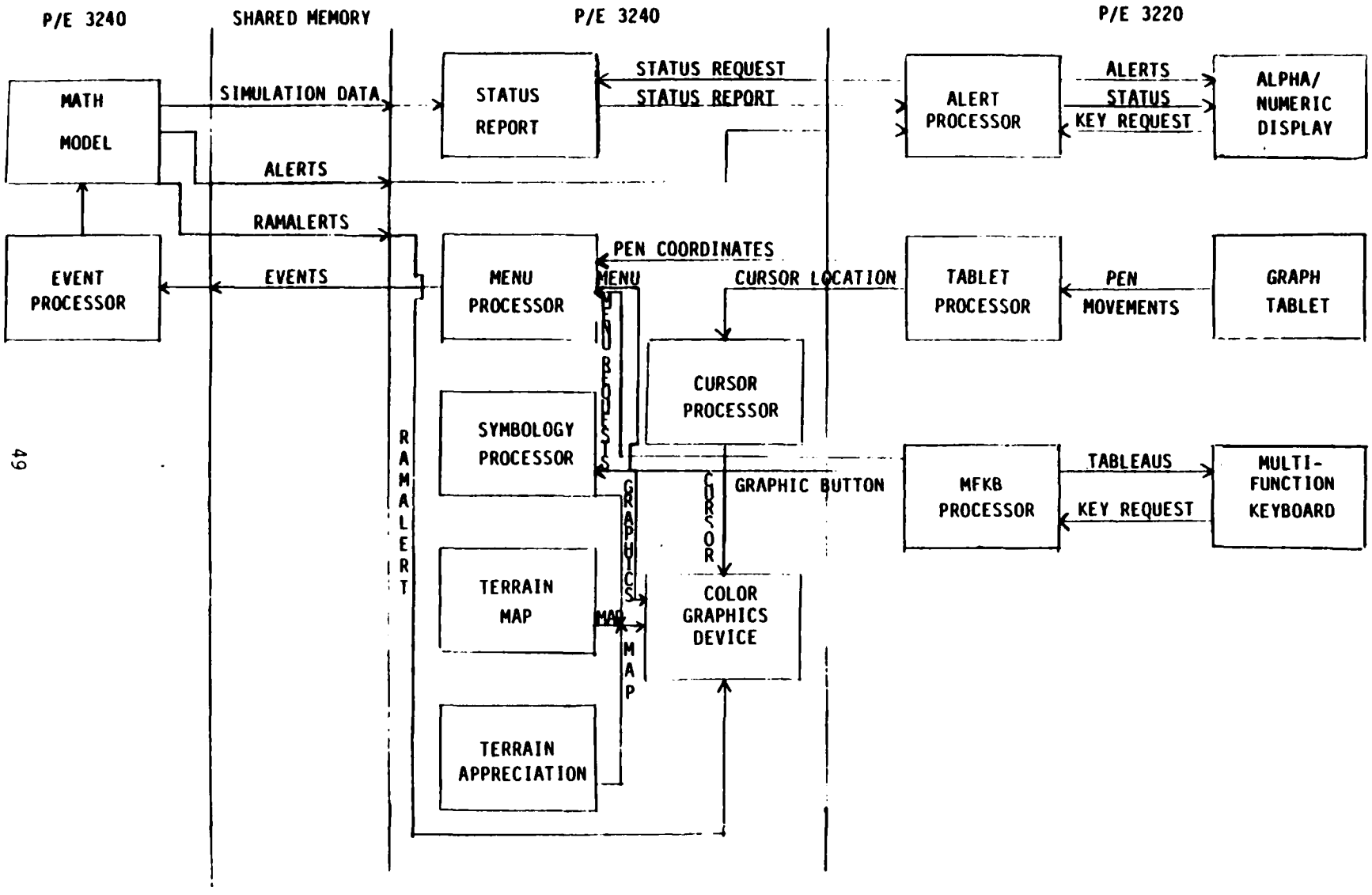


Figure 14. ARTBASS-M equipment connection.

The interaction of the VAX 11/780 containing the math model and the front-end system for each control station is shown in Figure 15.

4.2.4 Mace.

Mace consists of six Corvus Concept microcomputers sharing a 20 megabyte hard disk. One Corvus microcomputer is the executive of the system, one is a graphics preprocessor and the other four are the processors for each control station. The control stations include : 2 maneuver stations, 1 admin/log station and 1 support fire station. The control stations and executive station include the following hardware:

1. 6 Corvus Concept microcomputer (512KB)
2. 1 Corvus hard disk (20MB)
3. 1 Omninet disk server
4. 1 Corvus 8" floppy drive
5. 4 Sony video disk players
6. 1 Sony large screen display
7. 3 Sony 19" monitors
8. 5 Okidata microline 82A printers
9. 1 64K Microfazer serial-to-serial printer buffer
10. 4 8K Microfazer serial-to-serial printer buffers
11. 1 1/2" video cassette recorder
12. 4 Joysticks with interface
13. 5 Mouse with interface

The interconnection of the Mace equipment is shown in Figure 16.

4.2.5 Hardware Assessment.

In order to provide common modeling capabilities for ARTBASS and NTC, it is necessary to have the man-machine interface hardware of both ARTBASS-M and NTC. The hardware should consist of one NTC control station for compatability testing and three ARTBASS-M stations for full model testing. The front-end hardware should consist of the following items:

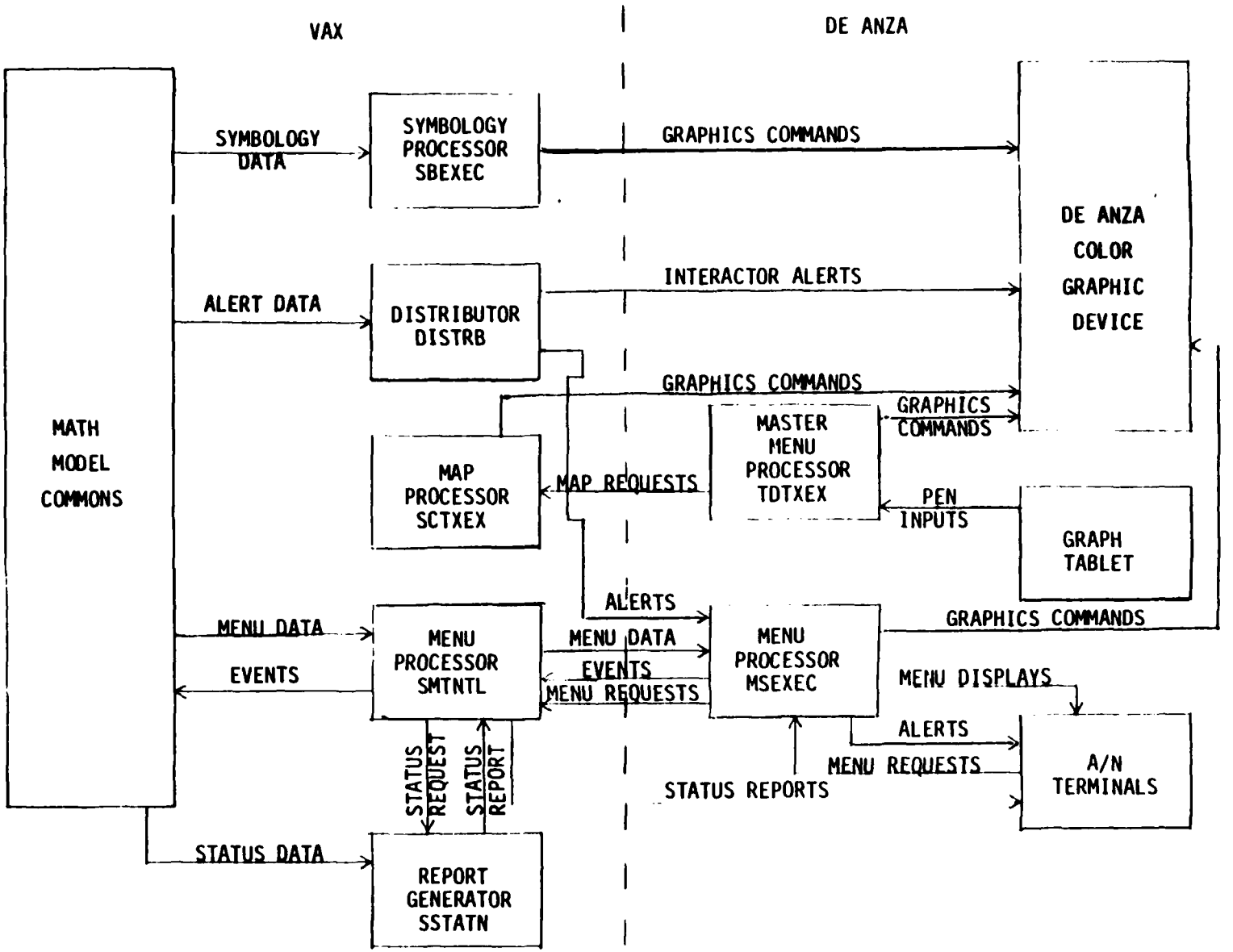


Figure 15. Test support dirver equipment connection.

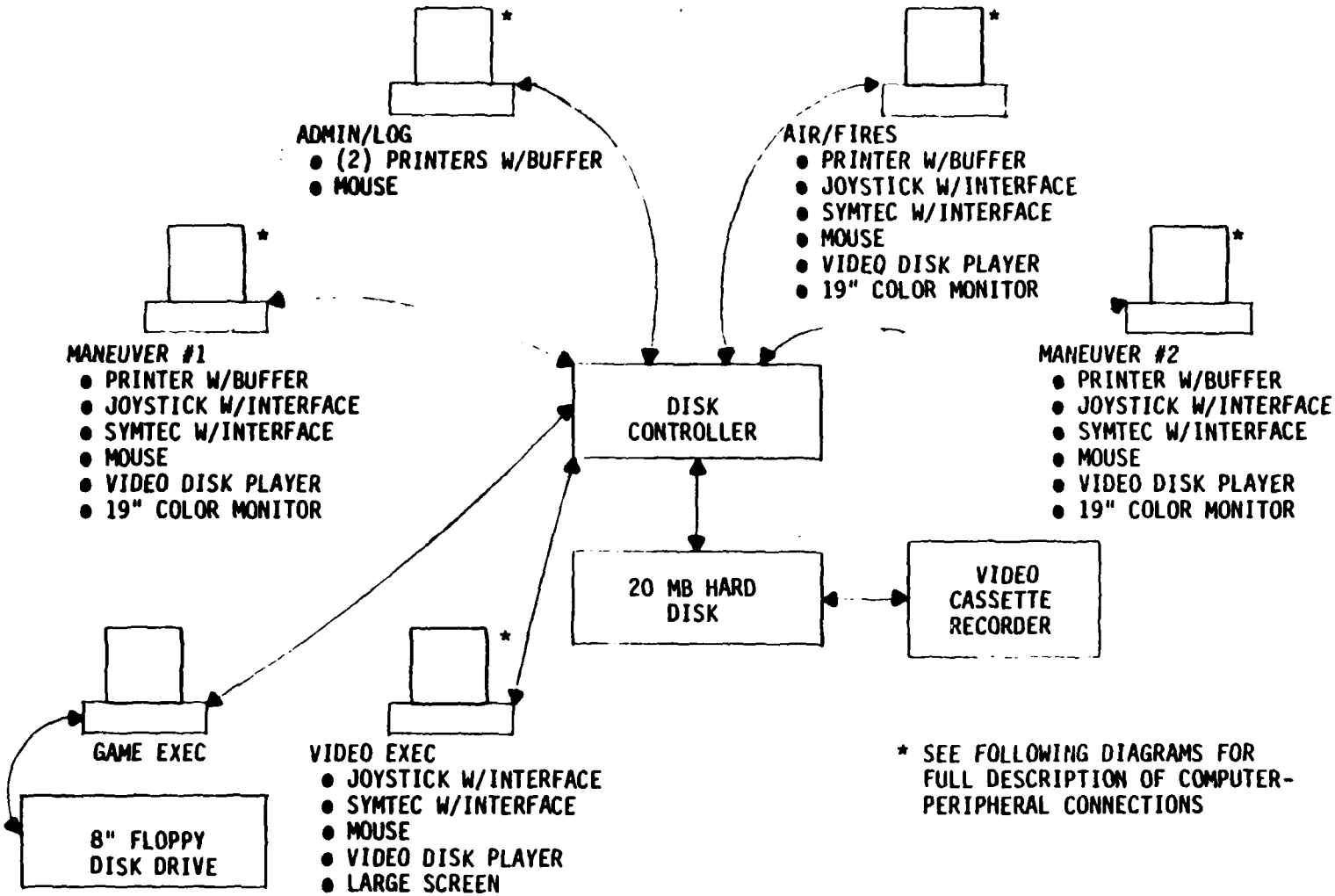


Figure 16. Mace equipment connection.

1. 1 De Anza VC23 color graphics processor

2. 2 VT-125 alphanumeric/graphic terminals

3. 1 Hitachi graph tablet and pen

4. 3 Lexidata 3400 color graphics processor

5. 3 Lexidata 19" color monitors

6. 3 Multifunction keyboards

7. 3 Summagraphics graph tablets and pens

8. 3 Perkin/Elmer OWEL 1251 alphanumeric terminals

9. 3 Control station printers

The interaction of the model and the De Anza is shown in Figure 17. and the interaction of the model and the Lexidatas is shown in Figure 18.

4.3 SIMULATION CONTROL.

The simulation control includes all controller actions necessary to start, stop, freeze or replay the model. Each system uses a different mechanism to interact with the controller for simulation control.

4.3.1 CATTS.

The CATTS math model is controlled by the simulation control switch on the control panel of the principal control station. Any of the three control stations can be identified as the principal controller at start up time. The control switch is used to step through initialization and to display the simulation control menu during the exercise. The simulation control menu provides the following functions:

1. Reinitialize

2. Replay

3. Restart

4. Terminate exercise

5. Freeze exercise

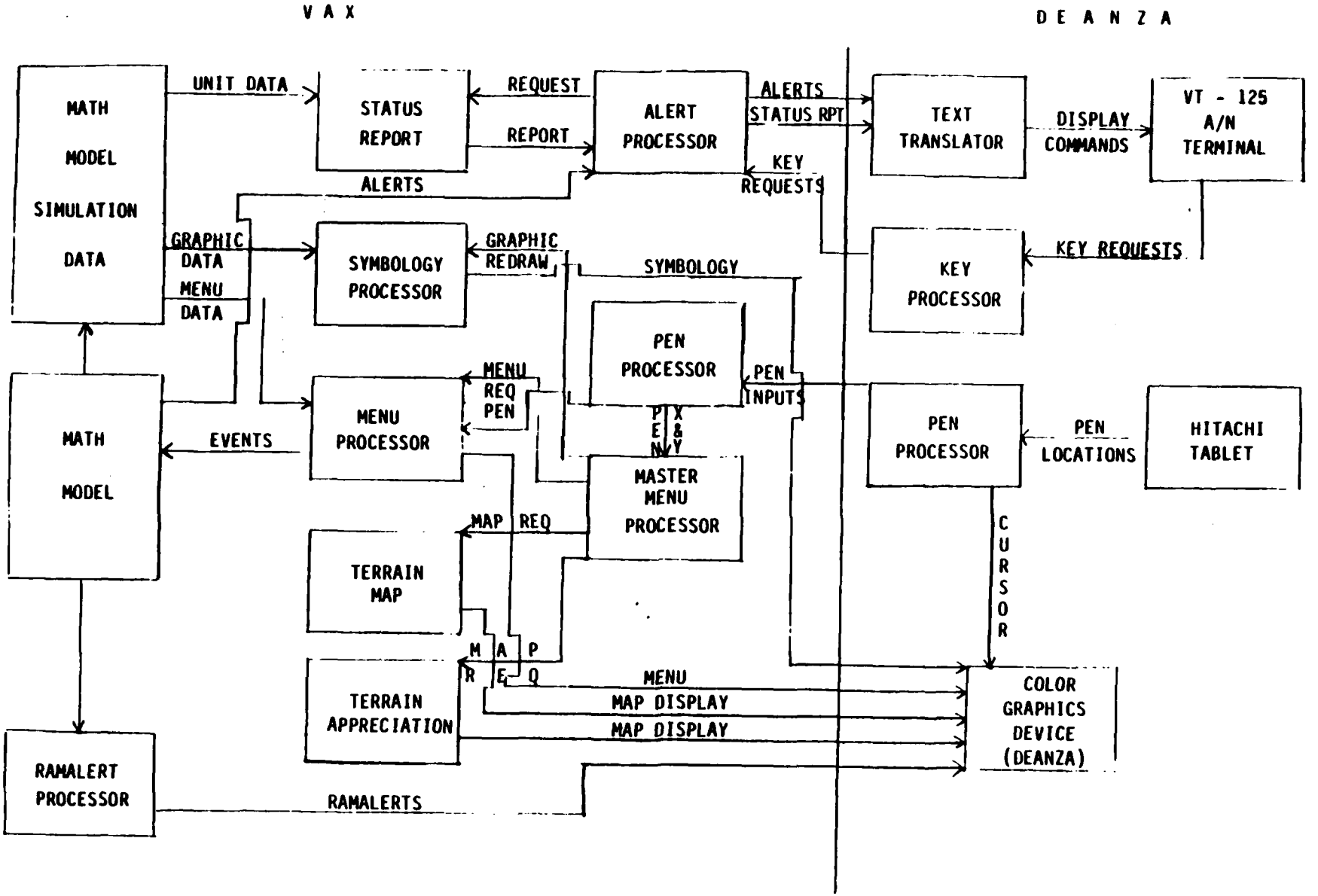


Figure 17. VAX/De Anza ARTBASS.

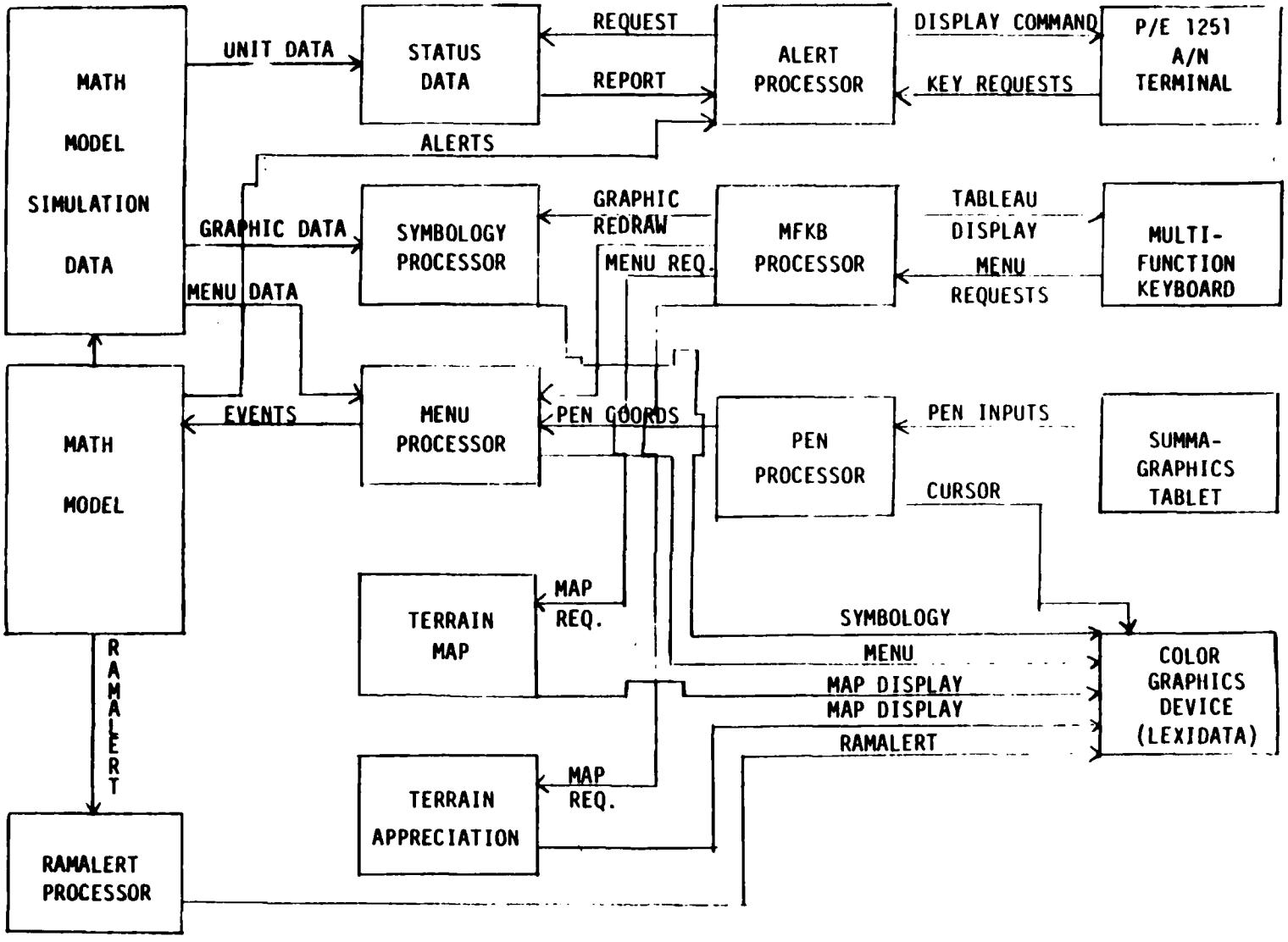


Figure 18. VAX/Lexidata ARTBASS.

4.3.2 ARTBASS-M.

The ARTBASS math model is controlled by the simulation control switch on the multifunction keyboard of the principal control station. Any of the five control stations can be identified as the principal controller at start up time. The control switch is used to step through initialization and to display the simulation control menu during the exercise. The simulation control menu provides the following functions:

1. Reinitialize
2. Replay
3. Restart
4. Terminate exercise
5. Freeze exercise

4.3.3 NTC Test Support Driver.

The Test Support Driver math model is controlled by the simulation control menu. The simulation control menu is displayed on the color monitor by selecting the simulation control button on the master menu (graph tablet). The menu is interacted with via the graph pen and color monitor and provides the following simulation controls:

1. Initialize scenario
2. Save initialization
3. Save exercise
4. Save command and control
5. Begin exercise
6. Reinitialize same scenario
7. Reinitialize new scenario
8. Reinitialize interactive initialization
9. Replay exercise
10. Restart the exercise

11. Terminate exercise

12. Terminate replicate

13. Terminate freeze

14. Freeze exercise

15. Replicate exercise

16. Produce end-of-game reports

The simulation control menu can only be executed from the principle control station. The principle control station is selected at start-up time.

4.3.4 Mace.

The Mace battle simulation is controlled by the Game Control module. The Game Control module resides in the Executive Control Station and is interacted with via an alphanumeric interactive menu.

4.3.5 Simulation Control Assessment.

The simulation control for the common model should be implemented in two ways. One should be identical to the ARTBASS simulation control using the multifunction keyboards on the three ARTBASS control stations. The other should be implemented using an interactive master menu on the De Anza and bit-pad for the NTC control station.

4.4 MAP DISPLAY.

The Map Display is the background map which is overlayed by the military symbology. The types of map displays described include video camera and map boards, 2-D digital maps, video disk images and 3-D digital maps.

4.4.1 CATTS.

The background map display is produced by a video camera aimed at a map board. Each control station is connected to a separate camera and has a separate map board. The camera position is controlled by a joy stick on the control panel of each control station. The joy stick provides the capability of moving the camera in any direction and zooming in or out.

The background map video signal is mixed with the graphic symbology signal produced by the Ramtek color graphics device and displayed on the color monitor.

4.4.2 ARTBASS-M.

The background map display for ARTBASS is a digital image of the exercise area. The digital map can be displayed with cross-country movement or vegetation background. Both backgound types are generated from DMA digitized terrain data and reflect the same terrain data that is used by the model for line of sight and cross-country movement speeds. The background maps can be overlayed by any or all of the following features:

1. Contour lines
2. Hydrography
3. Lines of communication
4. Grid lines
5. Labels

In addition to the terrain map, a terrain appreciation map is available on three control stations. The terrain appreciation presents a three-dimensional representation of the terrain from a controller selectable location and elevation.

The terrain map and the terrain appreciation are selected via a combination of inputs through the multifunction keyboard and the graph tablet.

4.4.3 NTC Test Support Driver.

The background map for the NTC Test Support Driver is a digital image of the exercise area. The digital map can be displayed with a cross-country movement background or a shaded relief background. The cross-country movement map is created from the same DMA data as the terrain database used by the cross-country movement module; and therefore, the map image displayed on the color monitor matches the terrain used by the model. The shaded relief map depicts the elevation of the terrain by shading the map image according to a user selectable sun angle. Both map backgrounds can be overlayed by any or all of the following features:

1. Contour lines
2. Built-up areas
3. Hydrography

4. Lines of communication

5. Sun position

6. Zoom control

7. Map position scroll

8. Grid lines

9. Miscellaneous features

Any of the above features can be displayed with or without the background map. The background maps are available in six display levels. The display levels and features are selected on the master menu using the graph pen and tablet.

4.4.4 Mace.

The background maps are stored as camera-produced images on video disk. The map location and zoom level are selected by the joy stick. The joy stick provides the capability to move in the X or Y direction and to zoom in or out. A total of six zoom levels are stored on the video disk.

The correct frame is retrieved from the video disk by converting the inputs from the joy stick to a frame number. The video signal from the video disk player is mixed with the graphic display signal by the Syntec PGS graphic device and displayed on the color monitor.

4.4.5 Map Display Assessment.

The map displays for the common model should be consistent with the ARTBASS and NTC graphic devices (Lexidata and De Anza). Since the interfacing with these devices is completely different, there should be two sets of map images and map display software. The ARTBASS map display software should be used to display the background maps for the three ARTBASS control stations and the NTC map display software should be used to display the background map on the NTC control station.

4.5 TACTICAL/OPERATIONAL MENUS.

The Tactical/Operational Menus provide the means by which the controller inputs commands to the model. Most of the interactive menus are displayed on the color monitor and selections are made using the graph tablet. However some of the menus are displayed and selections are made using an alphanumeric terminal or multifunction keyboard.

4.5.1 CATTS.

Tactical/operational menus are initiated by pressing the button on the control panel that corresponds to the desired action. The menu is displayed on the color monitor and is manipulated using the graph pen. CATTS provides the following interactive menus:

1. Activate units
2. Unit location
3. Maneuver control
4. Support fire
5. Direct fire
6. Air mission
7. Air defense
8. Preplanned mission
9. Control measures
10. Resupply
11. Weather
12. Task organization

4.5.2 ARTBASS-M.

Tactical/operational menus are initiated by selecting the desired menu from the multifunction keyboard. The menu is displayed on the color monitor and is interacted with via the graph tablet and pen. The type of units displayed on the menu is dependent upon button settings on the multifunction keyboard. A menu can be ignored from the graph tablet or the multifunction keyboard. ARTBASS provides the following interactive menus:

1. Activate units
2. Unit location
3. Maneuver control
4. Support fire

5. Direct fire

6. Air mission

7. Air defense

8. Preplanned mission

9. Control measures

10. Resupply

11. Weather

12. Alert routing

13. Significant event

4.5.3 NTC Test Support Driver.

Tactical/operational menus are initiated by touching the button on the master menu with the graph pen that corresponds to the desired menu. The menu is displayed on the color monitor and is interacted with via the graph pen and tablet. TSD provides the following interactive menus:

1. Activate units

2. Unit location

3. Maneuver control

4. Support fire

5. Direct fire

6. Air mission

7. Preplanned mission

8. Obstacle

9. Control measure

10. Intelligence control

11. Resupply

12. Weather

13. Unit bin definition

14. Task organization

4.5.4 Mace.

The Mace interactive menus are displayed on the alphanumeric display and are interacted with via the keyboard. The menus available to a controller is dependent upon the type of control station. The following list shows the menus that are available at each control station:

1. Executive
    - Control measures / obstacles
    - Initialize units
    - Set time
    - Simulation control

2. Maneuver (1 and 2)
    - Unit maneuver
    - Air maneuver
    - Unit engagement

3. Air/Fire
    - Artillery fire
    - Air fire

4. Admin/Log
    - Resupply
    - Status reports
    - Assessment reports

4.5.5 Menu Assessment.

To provide the capability to make changes to an interactive menu and have it be included on both the ARTBASS and NTC

station, only one menu system should exist. To stay compatible with the Perkin-Elmer ARTBASS system, the ARTBASS menu system is the only reasonable choice.

The Perkin-Elmer menu software can be directly converted to work on the VAX ARTBASS control stations; however, the NTC control station graphics processor has a horizontal pixel resolution of 512 compared with 640 for the ARTBASS graphics processor. To display the ARTBASS menus on the NTC control station, the following changes must be made:

1. The Lexidata graphic display utilities must be rewritten to work on the De Anza.

2. The time portion of the ARTBASS menu must be removed when displayed on the NTC control station. If the default time is to be changed, it can be selected along with the done, repeat, and ignore commands.

3. The rest of the interactive menu must be scaled down by the graphic utilities to fit into 512 pixels.

The interactive menus will be initiated from the multifunction keyboard on the ARTBASS stations and from the master menu on the NTC control station.

4.6 SYMBOLOGY.

Symbology includes all graphic overlays on the map display. It includes such items as unit locations, tactical overviews, impacting fires and control measures.

4.6.1 CATTS.

The symbology is displayed on the color monitor once every time-step for ground units and once every 15 seconds for air units. The symbology is selected by pressing the buttons corresponding to the desired symbols. The blue force symbology is displayed in blue and the red force symbology is displayed in red. The graphic display includes:

1. Unit direction of movement

2. Control measures

3. Engagement vectors

4. Air missions

5. Impacting fires
6. Smoke
7. Illumination
8. FEBA
9. Minefields
10. Obstacles and fortifications
11. Weapon systems
12. Sensors
13. Sensor coverage
14. Visually detected enemy units
15. Command posts

4.6.2 ARTBASS-M.

The symbology is displayed on the color monitor once every time-step for ground units and once every 15 seconds for air units. The symbology is selected from a menu on the multifunction keyboard. The blue force symbology is displayed in blue and the red force symbology is displayed in red. The graphic display includes:

1. Unit direction of movement
2. Control measures
3. Engagement vectors
4. Air missions
5. Impacting fires
6. Smoke
7. Illumination
8. FEBA
9. Minefields
10. Obstacles and fortifications

11. Weapon systems

12. Sensors

13. Sensor coverage

14. Visually detected enemy units

15. Command posts

4.6.3 NTC Test Support Driver.

The symbology is displayed on the color monitor once every time-step for ground units and once every 15 seconds for air units. The symbology is also refreshed when the digital map location or display level is changed. The control of which symbology appears on the monitor is accomplished by selecting the master menu buttons that correspond to the desired symbology. The buttons are selected via the graph pen and tablet.

The blue force symbology is displayed in blue and the red force symbology is displayed in red. The symbology graphic display includes:

1. Unit direction of movement

2. Control measures

3. Engagement vectors

4. Air missions

5. Impacting fires

6. Smoke

7. Illumination

8. FEBA

9. Minefields

10. Obstacles and fortifications

11. Local weather cells

12. International boundaries

13. Weapon systems

14. Sensors

15. Sensor coverage

16. Visually detected enemy units

17. Command posts

4.6.4 Mace.

The Mace symbology is displayed by the Syntec PGS graphic device. The selection of symbology to be displayed is accomplished by an interactive menu on each control station that has a color monitor. The following symbology can be displayed:

1. Unit (area occupied; iconic display; tactical overview; opcode)

2. Firing lines

3. Impacting fires

4. Air strikes

5. Control measures

6. Obstacles

The symbology for the blue units is displayed in blue and the symbology for the red units is displayed in red.

4.6.5 Symbology Assessment.

To conform with the Perkin-Elmer ARTBASS, the ARTBASS graphic display software should be used for the common model on the VAX. The graphic software should not require any changes for display on the ARTBASS control stations; however, the graphic utilities and the map to pixel conversion routines must be rewritten to work on the NTC control station.

The graphic symbology selection will be done on the master menu for the NTC control station and on the multifunction keyboard on the ARTBASS control station.

4.7 SIDE PANEL DISPLAYS.

Side panel displays inform the interactor of current model time, map attributes, and model status. The side panel displays are displayed on sections of the color monitor not used by the map and symbology.

4.7.1 CATTS.

CATTS does not have side panel displays on the color monitor. The only item displayed on the color monitor that fits into the side panel category is the model time, which is displayed in the upper left corner of the monitor.

4.7.2 ARTBASS-M.

ARTBASS displays the current model time as an overlay of the digital map. The attributes of the displayed digital map are displayed on the free space on the bottom of the screen.

4.7.3 NTC Test Support Driver.

The side panels of the color monitor contain information indicating the status of the model and system. The side panel information includes:

1. Current simulation time
2. Map center UTM coordinates
3. Cursor UTM coordinates
4. Map attributes
5. Map display and zoom level
6. Master menu prompts
7. Color dictionary

4.7.4 Mace.

Mace does not provide side panel displays on the color monitor. The only item displayed on the color monitor that fits into the side panel category is the time, which is displayed on top of the background map.

4.7.5 Side Panel Assessment.

Because of the difference in resolution of the Lexidata and De Anza, the side-panel information must be displayed in different screen locations. The De Anza has free space on both sides of the map display; therefore, the map attributes should be displayed on the screen sides for the NTC station. The Lexidata has free space on the bottom of the map display; therefore, the map attributes should be displayed on the screen bottom for the ARTBASS stations.

4.8 ALPHANUMERIC DISPLAYS.

The alphanumeric displays are used to keep the controller informed of what is happening in the model. Alerts of significant happenings in the model are displayed on the CRT and current unit status reports can be displayed upon controller request.

4.8.1 CATTS.

4.8.1.1 Unit Special Status Report. The unit special status report is displayed on the alphanumeric display when the status report function key is hit. The unit is selected by entering the unit name or number. The following unit information is displayed in the status report:

1. Simulation time of the report
2. Unit name
3. Unit number
4. Unit UTM location
5. Unit operational state
6. Unit rate of movement
7. Unit altitude
8. Unit suppression level percent
9. Units surrounding vegetation class
10. Visual detected equipment
11. Ammunition current load
12. Equipment initial and current load and number manned
13. Personnel initial and current level
14. Fuel current load

4.8.1.2 Log/Admin Status Report. The log/admin report is included is the unit status report.

4.8.1.3 Tactical alerts. The math model generates tactical alerts indicating significant unit events. The events include visual detections, engagements, rate of movement changes, obstacle encounters, and losses. The alerts are

assigned to a console(s) by unit. The alert routing can be updated by the task organization menu.

A tactical alert can be printed, routed to another console with an attached message, or saved to be looked at in the future. Alerts are dropped from the display by hitting the drop function key or a whole page of alerts can be dropped by hitting the page drop function key.

4.8.1.4 Interactor Alerts. Interactor alerts are displayed on the color monitor. The alerts identify menu errors input by the interactor and indicate if the menu was accepted or not.

4.8.2 ARTBASS-M.

4.8.2.1 Unit Special Status Report. The unit special status report is displayed on the alphanumeric display when the status report function key is hit. The unit is selected by entering the unit name or number. The following unit information is displayed in the status report:

1. Simulation time of the report
2. Unit name
3. Unit number
4. Unit UTM location
5. Unit operational state
6. Unit rate of movement
7. Unit altitude
8. Unit suppression level percent
9. Units surrounding vegetation class
10. Visual detected equipment
11. Ammunition current load
12. Equipment initial and current load and number manned
13. Personnel initial and current level
14. Fuel current load

If all the unit data cannot be displayed on one screen, the rest of the data can be displayed by hitting the page drop key.

4.8.2.2 Log/Admin Status Report. The log/admin report is included in the unit status report.

4.8.2.3 Tactical alerts. The math model generates tactical alerts indicating significant unit events. The events include visual detections, engagements, rate of movement changes, obstacle encounters and losses. The alerts are assigned to a console(s) by unit. The alert routing can be updated by the alert routing menu.

A tactical alert can be printed, routed to another console with an attached message or saved to be looked at in the future. Alerts are dropped from the display by hitting the drop function key or a whole page of alerts can be dropped by hitting the page drop function key.

4.8.2.4 Interactor Alerts. Interactor alerts are displayed on the color monitor and on the multifunction keyboard. The alerts identify menu input errors by the interactor and indicate if the menu was accepted or not.

4.8.3 NTC Test Support Driver.

4.8.3.1 Unit Special Status Report. The unit special status report is displayed on one of the two alphanumeric displays when the unit status function key is hit. The unit is selected by entering the unit name. The following information is displayed on the unit status report:

1. Simulation time of report
2. Unit name
3. Unit number
4. Unit UTM location
5. Unit operational state
6. Unit rate of movement
7. Unit altitude
8. Unit suppression level percent
9. Units surrounding vegetation class

10. Visual detected equipment

4.8.3.2 Log/Admin Status Report. The log/admin status report is displayed on one of the two alphanumeric displays when the log/admin function key is hit. The unit whose assets are to be displayed on the screen is selected by entering the unit name. The log/admin report includes:

1. Simulation time of report
2. Unit name
3. Unit number
4. Unit UTM location
5. Unit operational state
6. Equipment initial and current load and number manned
7. Ammunition current load
8. Personnel initial and current level
9. Fuel current load

4.8.3.3 Tactical Alerts. The TSD math model generates tactical alerts indicating significant unit and exercise events. The events include visual detections, engagements, rate of movement changes, obstacle encounters, losses, and exercise status. The alerts to be displayed are selected by each controller by alert category and unit via an interactive menu on the alphanumeric display.

The tactical alerts are automatically scrolled on the alphanumeric display and may be printed on the console printer.

4.8.3.4 Interactor Alerts. Interactor alerts, indicating interactor menu input errors and game status, are displayed on the color monitor. The alert is erased from the screen by hitting the ignore section of the alert with the graph pen.

4.8.4 Mace.

4.8.4.1 Unit Special Status Report. A unit special status report is available on the ALOG console to show the current status of a selected unit. The status report is selected via a menu and the unit is selected by entering the unit opcode.

4.8.4.2 Log/Admin Status Report. A log/admin status report is available on the ALOG console to show the current level of personnel and equipment for a selected unit. The report is selected via a menu and the unit is selected by entering the unit opcode.

4.8.4.3 Tactical Alerts. Instead of tactical alerts, Mace provides loss reports which are displayed at the ALOG console. The loss reports can be printed and delivered to the appropriate controller.

4.8.4.4 Interactor Alerts. Interactor alerts do not exist in Mace.

4.8.5 Alphanumeric Display Assessment.

To avoid complications in making changes to alerts or status reports, only one alert and status system should be implemented into the common model. Being it is necessary to conform to the Perkin-Elmer ARTBASS system, the ARTBASS alert and status report system should be implemented on both types of control stations. The implementation of the ARTBASS alert and status system should not require any changes to work on the ARTBASS control stations attached to the VAX. However, for the alerts and status reports to work on the NTC control station, the low level routines containing the terminal I/O commands must be rewritten to interact with the VT-125 instead of the P/E 1251.

DISTRIBUTION LIST

DEPARTMENT OF DEFENSE

Armed Forces Staff College
ATTN: Library

Assist to the Sec of Def, Atomic Energy
ATTN: Mil Appl, C. Field
ATTN: R. Wagner

Defense Advanced Rsch Proj Agency
ATTN: TTO

Defense Intell Agency
ATTN: Library
ATTN: RTS-2B

Defense Nuclear Agency
ATTN: NASF
ATTN: NATF
ATTN: NAWE
ATTN: RAAE
ATTN: RAAE, K. Schwartz
ATTN: RAEE
ATTN: RAEV
ATTN: SPSS
ATTN: SPTD
ATTN: STBE
ATTN: STNA
ATTN: STRA
ATTN: STSP
4 cys ATTN: STTI-CA

Defense Tech Info Center
12 cys ATTN: DD

Dep Under Sec of Def
ATTN: S&TNF, T. Jones

Field Command, DNA, Det 2
Lawrence Livermore National Lab
ATTN: FC-1

DNA PACOM Liaison Ofc
ATTN: J. Bartlett

Field Command, Defense Nuclear Agency
ATTN: FCPRW
ATTN: FCTT, W. Summa
ATTN: FCTXE

Interservice Nuc Wpns School
ATTN: Doc Control

Joint Chiefs of Staff
ATTN: J-3, Strat Opns Div
ATTN: J-5, Nuc/Chem Plcy Br, J. Steckler
ATTN: J-5, Nuc Div/Strat Div
ATTN: J-5, Strat Div, W. McClain
ATTN: JAD/SFD
ATTN: JAD/SSD

National Defense University
ATTN: NWCLB-CR

Ofc of the Sec of Def, Net Assessments
ATTN: Doc Control

DEPARTMENT OF DEFENSE (Continued)

Principal Dep Under Sec of Def, Rsch & Engrg
ATTN: J. Wade Jr.

Program Analysis & Evaluation
ATTN: S. Johnson
ATTN: Strat Programs

US European Command
ATTN: ECJ-3
ATTN: ECJ-5

US Natl Mil Representative, SHAPE
Attention US Doc Ofc for
ATTN: Nuc Plans
ATTN: Intel
ATTN: Pol, Nuc Concepts

US Readiness Command
ATTN: J-3

Under Sec of Def for Policy
ATTN: Dir Plng & Requirements, M. Sheridan

Under Secy of Def for Rsch & Engrg
ATTN: K. Hinman

United States Central Command
ATTN: CCJ3-OX, Daigneault

DEPARTMENT OF THE ARMY

Asst Ch of Staff for Intell
ATTN: DAMI-FIT

Chemical Rsch & Dev Ctr
ATTN: SMCCR-OPR

Dep Ch of Staff for Ops & Plans
ATTN: DAMO-NCN
ATTN: DAMO-RQA, Firepower Div
ATTN: DAMO-RQS
ATTN: DAMO-SSM, Pol-Mil Div
ATTN: Tech Advisor
5 cys ATTN: DAMO-NC, Nuc Chem Dir

National Training Ctr
ATTN: TAF-NBC

US Army Armament Rsch Dev & Cmd
ATTN: DRDAR-LCN-E

US Army Ballistic Rsch Lab
ATTN: DRDAR-BLA-S, Tech Lib
ATTN: DRDAR-BLV
ATTN: R. Reisler

US Army Chemical School
ATTN: ATZM-CM-F
ATTN: ATZN-CM-CC
ATTN: ATZN-CM-N

US Army Comd & General Staff College
ATTN: DTAC
3 cys ATTN: Combined Arms Rsch Lib

DEPARTMENT OF THE ARMY (Continued)

US Army Comb Arms Combat Dev Acty
ATTN: ATZL-CAP-DT
ATTN: ATZL-SWN
ATTN: ATZL-SWP
ATTN: ATZL-SWT
ATTN: ATZL-TAS-S

US Army Concepts Analysis Agency
ATTN: CSSA-ADL, Tech Lib

US Army Engineer School
ATTN: Library

US Army Europe & Seventh Army
ATTN: AEAGC-NC-C

US Army Forces Command
ATTN: AF-OPTS
ATTN: AFOP-TN

US Army Foreign Science & Tech Ctr
ATTN: DRXST-SD-1

US Army Infantry Ctr & Sch
ATTN: ATSH-CD-CSO

US Army Intel Threat Analysis Det
ATTN: AIAIT-HI

US Army Intell Ctr & School
ATTN: ATSI-CD-CS

US Army Logistics Ctr
ATTN: ATCL-OOL, S. Cockrell

US Army Material Command
ATTN: DRCDE-D

US Army Materiel Sys Analysis Actvy
ATTN: X5, W3JCAA

US Army Mobility Equip R&D Cmd
ATTN: DRDME-WC, Tech Lib, Vault

US Army Nuclear & Chemical Agency
ATTN: Library
ATTN: MONA-CM
ATTN: MONA-NW
ATTN: MONA-OPS
ATTN: MONA-OPS, B. Thomas
ATTN: MONA-OPS, J. Ratway

US Army TRADOC Sys Analysis Actvy
ATTN: ATAA-TAC
ATTN: ATOR-TDB

US Army Training & Doctrine Comd
ATTN: ATCD-FA
ATTN: ATCD-N
ATTN: ATIC-NC

US Army War College
ATTN: AWCAC, F. Braden, Dept of Tactics
ATTN: Library
ATTN: War Gaming Facility

US Army Comb Arms Opns Rsch Acty
ATTN: ATOR-CAT-T

DEPARTMENT OF THE ARMY (Continued)

USA Military Academy
ATTN: Doc Lib

USA Missile Command
ATTN: DRSMI-RH
ATTN: DRSMI-XF

V Corps
ATTN: G-2
ATTN: G-3

VII Corps
ATTN: G-2
ATTN: G-3

DEPARTMENT OF THE NAVY

Marine Corps
ATTN: Code OTOO-31
ATTN: DCS, P&O, Requirements Div
ATTN: DCS, P&O, Strat Plans Div

Marine Corps Dev & Education Command
ATTN: Commander

Naval Postgraduate School
ATTN: Code 1424, Library

Naval Research Laboratory
ATTN: Code 2627, Tech Lib

Naval War College
ATTN: Code E-11, Tech Svc

Nuclear Weapons Tng Gp, Atlantic
ATTN: Nuclear Warfare Dept

Nuclear Weapons Tng Gp, Pacific
ATTN: Nuc Warfare Dept

DEPARTMENT OF THE AIR FORCE

Air Force Operational Test & Eval Ctr
ATTN: OA

Air University Library
ATTN: AUL-LSE

Assist Ch of Staff, Studies & Analysis
2 cys ATTN: AF/SAMI, Tech Info Div

Dep Ch of Staff, Plans & Opns
ATTN: AFXOOR, Opns, Opnl Spt

Foreign Technology Div
ATTN: SD
ATTN: TQ

DEPARTMENT OF ENERGY AGENCY

Sandia National Laboratories
ATTN: Tech Lib, 3141

DEPARTMENT OF DEFENSE CONTRACTORS

Kaman Tempo
ATTN: C. Anderson
ATTN: DASIAC

DEPARTMENT OF DEFENSE CONTRACTORS (Continued)

Science Applications International Corp
ATTN: B. Packard
ATTN: D. Erickson
ATTN: J. Birney
ATTN: J. Ickler
ATTN: J. Martin
ATTN: L. Metzger
ATTN: M. Drake
ATTN: P. McKeown
ATTN: R. Plock

DEPARTMENT OF DEFENSE CONTRACTORS (Continued)

Kaman Tempo
ATTN: DASIAC

END

FILMED

5-86

DTIC